Databases II

Language Technology and Web Applications

Jannis Vamvas

Department of Computational Linguistics
University of Zurich

October 23, 2024

University of Zurich[UZH]

Topics

1. SQL Queries

2. Database Optimization

3. Introduction to Server-Side Frameworks

Repetition Quiz

https://t.uzh.ch/1BW

Learning Goals for this Week

- You can perform basic **SQL queries** (with the help of documentation)
- You are aware of the concepts of **(de-)normalization** and **indices**
- You can describe the motivation for using **server-side frameworks** in complex web applications

Topics

1. SQL Queries

2. Database Optimization

3. Introduction to Server-Side Frameworks

Textbook

The following is a review of the most important concepts introduced in the PostgreSQL Tutorial.

(`https://www.postgresql.org/docs/current/tutorial.html`)

SELECT Statement

```sql
SELECT *
FROM customers;
```

```sql
SELECT name, birth_date
FROM customers;
```

WHERE Clause

```
SELECT title
FROM library
WHERE pub_year = 2017;
```

AND and OR Operators

```sql
SELECT name
FROM customers
WHERE canton = 'ZH'
    AND birth_year < 2003;
```

```sql
SELECT name
FROM customers
WHERE canton = 'BS'
    OR canton = 'BL';
```

NULL Values

```sql
SELECT phone_number
FROM customers
WHERE phone_number IS NOT NULL;
```

AS Clause

```
SELECT name AS movie_title
FROM movies;
```

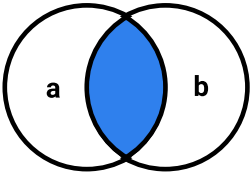
ORDER BY Clause

```sql
SELECT *
FROM customers
ORDER BY birth_date DESC;
```

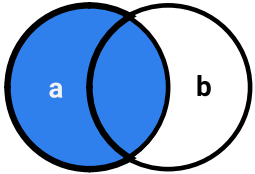
DISTINCT Clause

```sql
SELECT DISTINCT city
FROM customers;
```

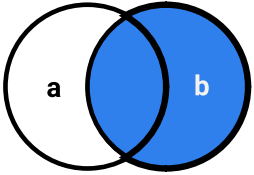
Joins



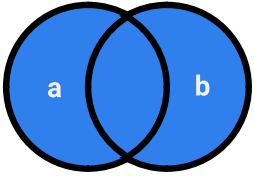
`a INNER JOIN b`



`a LEFT JOIN b`



`a RIGHT JOIN b`



`a FULL OUTER JOIN b`

Inner Joins

```
SELECT scientists.name, prizes.name
FROM scientists
INNER JOIN prizes
    ON prizes.id = scientists.prize_id;
```

COUNT() Aggregate Function

```sql
SELECT COUNT(*)
FROM books
WHERE in_stock;
```

SUM() and AVG() Aggregate Functions

```sql
SELECT SUM(price)
FROM orders;
```

```sql
SELECT AVG(price)
FROM orders;
```

MIN() and MAX() Aggregate Functions

```sql
SELECT MIN(price)
FROM orders;
```

```sql
SELECT MAX(price)
FROM orders;
```

GROUP BY Clause

```sql
SELECT rating, COUNT(*)
FROM movies
GROUP BY rating;
```

LIKE Operator

```sql
SELECT name
FROM movies
WHERE name LIKE 'The %';
```

```sql
SELECT name
FROM movies
WHERE name LIKE '_ove';
```

Tips for Advanced String Searching

Regex Search

- ~ operator
- https://www.postgresql.org/docs/current/functions-matching.html#FUNCTIONS-POSIX-REGEXP

Fuzzy String Matching

- pg_trgm module
- https://www.postgresql.org/docs/13/pgtrgm.html

Topics

1. SQL Queries

2. Database Optimization

3. Introduction to Server-Side Frameworks

Normalization

Definition

Database Normalization means to reduce redundancy in a database by separating the data into multiple tables.

- \+ Makes the database easier to maintain
- − In some cases, queries might become more expensive

Constraints

```sql
CREATE TABLE book
  (
     id      INT  PRIMARY KEY,
     title   TEXT NOT NULL,
     isbn    TEXT UNIQUE
  );
```

EXPLAIN ANALYZE

```
EXPLAIN ANALYZE
SELECT *
FROM table1 t1, table2 t2
WHERE t1.unique1 < 10 AND t1.unique2 = t2.unique2;
```

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.65..118.62 rows=10 width=488)
   ->  Bitmap Heap Scan on table1 t1  (cost=4.36..39.47 rows=10 width=244)
         Recheck Cond: (unique1 < 10)
         ->  Bitmap Index Scan on table1_unique1  (cost=0.00..4.36 rows=10 width=0)
               Index Cond: (unique1 < 10)
   ->  Index Scan using table2_unique2 on table2 t2  (cost=0.29..7.91 rows=1 width=244)
         Index Cond: (unique2 = t1.unique2)
 Planning time: 0.181 ms
 Execution time: 0.501 ms
```

Creating an Index

```sql
CREATE INDEX books_title_idx ON books(title);
```

Discussion of Indices

- \+ Makes searching and filtering a lot faster
- − Takes up additional space
- − Makes write operations slower

Topics

1. SQL Queries

2. Database Optimization

3. Introduction to Server-Side Frameworks

Motivation: Revisiting the Example Project

`https://t.uzh.ch/1p0`

- Find 5 things that could be done better/more elegantly in the Python code
- Bonus: Can you find the security vulnerability?

Object-Relational Mapping (ORM)

```
class Book:

    def __init__(self, title):
        self.title = title

    @classmethod
    def get_all_books(cls):
        sql = "SELECT title FROM books;"
        result = ...
        return [Book(title) for title in result]
```

Django: Models

```python
# models.py

from django.db import models


class Book(models.Model):
    title = models.TextField()
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
    year = models.IntegerField()

class Author(models.Model):
    first_name = models.TextField()
    last_name = models.TextField()
```

Django: QuerySets

```
from models import Book, Author


all_books = Book.objects.all()
recent_books = Book.objects.filter(year=2024)

my_books = Book.objects.filter(author__first_name="Jannis")
```

Django: Templates

```
<!-- index.html -->

<p>Hello, {{ username }}!</p>

<p>Books on my nightstand:
    <ul>
    {% for book in books %}
        <li>{{ book.title }}</li>
    {% endfor %}
    </ul>
</p>
```

Django: Views

```
from django.shortcuts import render

from models import Book


def index(request):
    context = {
        'username': request.POST['username'],
        'books': Book.objects.all()
    }
    return render(request, 'index.html', context)
```

Starting the Server using GitLab CI

Put the command into `Procfile`

Previously:

```
web: python server.py
```

Django:

```
web: python manage.py runserver 0.0.0.0:5000
```

Flask:

```
web: flask run --port 5000
```

Further Reading

- https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks
- https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django

Reminder: Take Home Exam

- Starts **25th October, 2024, at noon**
- Ends 1st November, 2024, at noon (one week later)
- Submit your solution via OLAT
- No lecture next week