

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»
(МТУСИ)

Кафедра «Программная инженерия»
дисциплина «Рефакторинг баз данных и приложений»

Лабораторная работа № 5
**«Рефакторинг приложений с целью оптимизации запросов и
индексов базы данных»**

Выполнил: студент группы БПИ2303
Берездовец Илья Сергеевич

Москва 2025

Цели лабораторной работы

1. Изучить основные принципы оптимизации SQL-запросов.
2. Научиться выявлять узкие места в работе базы данных.
3. Освоить создание, модификацию и удаление индексов для ускорения запросов.
4. Применить знания на практике для оптимизации базы данных и приложения.

Ход работы:

Изначальный код:

```
11 def get_db_connection():
12     db_url = os.getenv('DATABASE_URL')
13     conn = psycopg2.connect(db_url)
14     create_table_query = '''
15     CREATE TABLE IF NOT EXISTS vacancies (
16         id VARCHAR(255) PRIMARY KEY,
17         title TEXT,
18         snippet TEXT,
19         requirement TEXT,
20         salary TEXT,
21         url TEXT
22     )
23     '''
24     cursor = conn.cursor()
25     cursor.execute(create_table_query)
26     conn.commit()
27     cursor.close()
28     return conn
29
30 def get_vacancies(profession, page=0, per_page=20):
31     url = "https://api.hh.ru/vacancies"
32     params = {
33         'text': profession,
34         'page': page,
35         'per_page': per_page
36     }
37     response = requests.get(url, params=params)
38     response.raise_for_status()
39     return response.json()
40
41 def clean_text(text):
42     if text is None:
43         return ''
44     cleaned_text = re.sub(r'<[>]+>', '', text)
45     return cleaned_text
46
47 def parse_vacancies(data):
48     vacancies = []
49     for item in data['items']:
50         salary = item['salary']
51         if salary:
52             if salary['currency'] != 'RUR' and salary['currency']:
53                 continue
54             salary_str = f"{salary['from']}-{salary['to']} {salary['currency']}" if salary['from'] and salary['to'] else salary['from'] if salary['from'] else salary['to']
55         else:
56             salary_str = 'Не указана'
57
58         vacancy = {
59             'id': item['id'],
60             'title': clean_text(item['name']),
61             'snippet': clean_text(item['snippet'].get('responsibility')),
62             'requirement': clean_text(item['snippet'].get('requirement')),
63             'salary': salary_str,
64             'url': item['alternate_url']
65         }
66         vacancies.append(vacancy)
67     return vacancies
68
69 def save_to_db(vacancies, conn):
70     cursor = conn.cursor()
71
72     insert_query = '''
73     INSERT INTO vacancies (id, title, snippet, requirement, salary, url)
74     VALUES (%s, %s, %s, %s, %s, %s)
75     ON CONFLICT (id) DO UPDATE SET
76     title = EXCLUDED.title,
77     snippet = EXCLUDED.snippet,
78     requirement = EXCLUDED.requirement,
79     salary = EXCLUDED.salary,
80     url = EXCLUDED.url
81     '''
82
83     for vacancy in vacancies:
84         cursor.execute(insert_query, (
85             vacancy['id'],
86             vacancy['title'],
87             vacancy['snippet'],
88             vacancy['requirement'],
89             vacancy['salary'],
90             vacancy['url']
91         ))
```

Рис. 1 – (первая часть исходного кода)

```

69 def save_to_db(vacancies, conn):
70     )
71
72     conn.commit()
73     cursor.close()
74
75 def salary_to_numeric(salary_str):
76     if isinstance(salary_str, int):
77         return salary_str
78     if salary_str == 'Не указана':
79         return 0
80
81     try:
82         parts = salary_str.split('-')
83         if len(parts) == 2:
84             return (int(parts[0].strip()) + int(parts[1].strip())) // 2
85         return int(parts[0].strip())
86     except (ValueError, IndexError):
87         return 0
88
89 @app.route('/', methods=['GET', 'POST'])
90 def index():
91     """
92     Main page to search for vacancies
93     """
94     parameters:
95     - name: profession
96       in: query
97       type: string
98       required: true
99       description: Profession to search for
100     - name: num_pages
101       in: query
102       type: integer
103       required: false
104       default: 5
105       description: Number of pages to fetch
106
107     responses:
108     200:
109       description: A list of vacancies
110       schema:
111         type: array
112         items:
113           type: object
114           properties:
115             id:
116               type: string
117             title:
118               type: string
119             snippet:
120               type: string
121             requirement:
122               type: string
123             salary:
124               type: string
125             url:
126               type: string
127
128     """
129     vacancies = []
130     profession = ''
131     num_vacancies = 0
132
133     if request.method == 'POST':
134         profession = request.form['profession']
135         num_pages = int(request.form.get('num_pages', 5))
136         conn = get_db_connection()
137
138         all_vacancies = []
139         for page in range(num_pages):
140             data = get_vacancies(profession, page=page)
141             vacancies_page = parse_vacancies(data)
142             all_vacancies.extend(vacancies_page)
143
144         save_to_db(all_vacancies, conn)
145         conn.close()
146
147         vacancies = all_vacancies
148         num_vacancies = len(vacancies)
149
150     return render_template('index.html', vacancies=vacancies, profession=profession, num_vacancies=num_vacancies)
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168

```

Рис. 2 – (вторая часть изначального кода)

```

261 cursor.execute('''
262 SELECT title, ROUND(AVG(
263     CASE
264         WHEN position('-', in salary) > 0 THEN
265             (CAST(substring(salary, '\d+)-') AS bigint) + CAST(substring(salary, '-(\d+)') AS bigint)) / 2
266         ELSE CAST(REGEXP_REPLACE(salary, '[^0-9]', '', 'g') AS bigint)
267     END
268 )) AS average_salary
269 FROM vacancies
270 WHERE salary <> 'Не указана' AND salary LIKE '%RUR%'
271 GROUP BY title
272 ORDER BY average_salary DESC
273 LIMIT 3
274 ''')
275 top_salaries = cursor.fetchall()
276
277 top_salaries = [{'name': row[0], 'average_salary': int(row[1])} for row in top_salaries]
278
279 cursor.close()
280 conn.close()
281 return render_template('all_vacancies.html', vacancies=vacancies, vacancy_count=vacancy_count, top_salaries=top_salaries)
282
283
284 if __name__ == '__main__':
285     app.run(host='0.0.0.0', port=5000, debug=True)
286

```

Рис. 3 – (третья часть изначального кода)

Применю следующие улучшения коду, а именно SQL – запросам.

1. Индексы: добавлю индекс на title и salary — часто используются в выборке и группировке.
2. Уберу SELECT *: заменю выборки на конкретные столбцы.
3. Оптимизирую SQL-запросы: исправлю выражения с substring, чтобы избежать ненужных операций.
4. Добавлю анализ через EXPLAIN.

1.

```

# Индексы — для ускорения WHERE, GROUP BY, ORDER BY
cursor.execute('CREATE INDEX IF NOT EXISTS idx_title ON vacancies(title)')
cursor.execute('CREATE INDEX IF NOT EXISTS idx_salary ON vacancies(salary)')

```

Рис. 4 - (Добавил индексы)ё

2.

```

cursor.execute('SELECT id, title, snippet, requirement, salary, url FROM vacancies')

```

Рис. 5 – (Добавил выборку на конкретные столбцы)

3.

```

IN salary LIKE '%%' THEN
    (CAST(split_part(salary, '-', 1) AS bigint) + CAST(split_part(split_part(salary, '-', 2), ' ', 1) AS bigint)) / 2
ELSE CAST(REGEXP_REPLACE(salary, '[^0-9]', '', 'g') AS bigint)

```

Рис. 6 – (Исправил substring на split_part)

4.

```

cursor.execute('EXPLAIN ANALYZE SELECT title, salary FROM vacancies WHERE salary <> 'Не указана' AND salary LIKE '%RUR%' ORDER BY title')

```

Рис. 7 – (Добавил анализ через EXPLAIN)

Резюмирую улучшения:

1. Добавил индексы на `title` и `salary`, так как они часто участвуют в фильтрации и группировке — это ускоряет запросы.

2. Заменял `SELECT *` на выбор конкретных столбцов — уменьшает нагрузку и повышает читаемость.
3. Оптимизировал SQL-запросы, переписал `substring`-выражения, чтобы убрать лишние операции.
4. Добавил `EXPLAIN ANALYZE` для анализа запросов и оценки их производительности.

Вывод:

1. Изучил основные принципы оптимизации SQL-запросов.
2. Научился выявлять узкие места в работе базы данных.
3. Освоил создание, модификацию и удаление индексов для ускорения запросов.
4. Применил знания на практике для оптимизации базы данных и приложения.

Ссылка на GitHub: <https://github.com/iberezaa/RBDiP>