PostgreSQL 与 JSON 实战

段清华

文因互联 – Memect

2016 年 6 月 26 日

JSON 格式

- JavaScript Object Notation (JSON) 数据交换格式， RFC 7158

JSON 格式

- JavaScript Object Notation (JSON) 数据交换格式， RFC 7158
- 优点：
  - 轻量级
  - 人类友好
  - 无处不在
  - 相对于 XML 来说，更简单，更紧凑

JSON 样例


```
{
    " 名称 ": " 华夫饼 ",
    " 人数 ": 2,
    " 口味 ": null,
    " 辅料 ": [" 牛奶 ", " 玉米淀粉 ", " 泡打粉 "],
    " 步骤 ": [
        {" 说明 ": " 首先 "},
        {" 说明 ": " 然后 "}
    ]
}
```

选择数据库

选择数据库

	SQL	文档数据库	图数据库
优点	擅长处理关系，提供各种表的合并，选择，筛选操作 产业成熟	有更多定制化功能，例如 map-reduce ，全文检索 查询方便，直观	方便进行数据推理，例如一道菜有多种材料组成，而其中一些材料是由其他材料组成，需要递归查询
缺点	传统 SQL 数据库表结构死板，修改成本高， PostgreSQL 一定程度上解决了这个问题	不擅长数据之间的联合查询（表 JOIN ） 相对没有 SQL 成熟 不同数据库差别大	使用成本高 当查询大的时候速度慢

结论（可能的）

- 需要更丰富的全文检索功能，允许文档间关系弱化?
  - Elasticsearch
- 需要更全面的推理与查询定制，允许高成本?
  - 图数据库， Neo4j / Stardog
- 我不知道需要什么?
  - 尝试下 PostgreSQL
    - 更成熟，更快速，更多支持

PostgreSQL 有什么?

PostgreSQL 有什么?

- 支持全文检索，可能需要手动分词，速度不慢，功能支持对比 ElasticSearch 少不少（例如 boost ）

PostgreSQL 有什么?

- 支持全文检索，可能需要手动分词，速度不慢，功能支持对比 ElasticSearch 少不少（例如 boost ）
- 推理查询，比较固定和死板，很容易导致 SQL 查询的迅速复杂化，优点是条例比较清晰

PostgreSQL 有什么?

- 支持全文检索，可能需要手动分词，速度不慢，功能支持对比 ElasticSearch 少不少（例如 boost ）
- 推理查询，比较固定和死板，很容易导致 SQL 查询的迅速复杂化，优点是条例比较清晰
- 丰富的 JSON 操作，甚至可以媲美一些文档数据库，但是为了保证 SQL 风格，相对文档数据库更不直观，学习成本相对更高

早期 PostgreSQL 与 JSON

- 9.2 版本之前：用文本存储 JSON （ text ）
- 9.2 版本，引入 json 类型，依然以文本存储，但是可以自动解析。提供了 row_to_json 和 array_to_json 函数
- 9.3 引入 to_json ， json_agg 函数，相对于 9.2 的函数来说速度上有了提高

- 此时缺点：支持不完善，缺乏处理功能

现在 PostgreSQL 与 JSON

- 9.4 引入 jsonb 类型
  - 提供更高效的操作
  - 允许索引
- 9.5 提供 jsonb_set， jsonb_pretty 等更多功能

JSON 与 JSONB

- JSON
  - 以文本方式存储 JSON
  - 读写速度快
  - JSON 的各种操作很慢
- JSONB
  - 以解析好的二进制存储 JSON
  - 读写因为需要解析，速度相对慢
  - 各种选择与查询操作更快
  - 如果没有极特殊需求，应该作为首选

准备 PostgreSQL

- 在 Ubuntu 下安装
  - # sudo apt-get install postgresql
- 切换到 postgres 用户（默认 PostgreSQL 用户）
  - # sudo su – postgres
- 使用 psql 对数据库进行简单管理
  - # psql

常用 psql 命令

- 查看连接信息
  - postgres=# \conninfo
- 查看数据库列表
  - postgres=# \l
- 进入 database
  - postgres=# \c food_db
- 查看表结构
  - food_db=# \d
- 退出
  - food_db=# \q

数据库创建

- 修改密码
  - # ALTER USER postgres PASSWORD 'newpass';
- 创建数据库
  - # CREATE DATABASE food_db;
- 分号很重要

PostgreSQL 与 Python

```
import psycopg2
conn = psycopg2.connect(
    database='food_db', host='localhost',
    port='5432', user='postgres', password='newpass')
cursor = conn.cursor()
cursor.execute('''
CREATE TABLE IF NOT EXISTS
dish (name text, data jsonb)
''')
conn.commit() #  插入、修改时务必
```

Stargazy Pie 仰望星空

Image from wiki:
https://en.wikipedia.org/wiki/Stargazy_pie

SQL 记录的增加

```
cursor.execute('''
INSERT INTO dish
VALUES (%s, %s)
''', [' 仰望星空 ', '{" 人数 ": 2, " 时间 ": "30 分钟 "}'])
conn.commit() #  插入、修改时务必
```

SQL 记录的查找

```
cursor.execute('''
SELECT name, data
FROM dish
WHERE name = %s
''', [' 仰望星空 '])
```

SQL 记录的修改

```
cursor.execute('''
UPDATE dish
SET name=%s, data=%s
WHERE name=%s
''', [' 仰望星空 ', '{" 人数 ": 2, " 时间 ": "30 分钟 "}',
' 仰望星空 '])
conn.commit() #  插入、修改时务必
```

SQL 记录的删除

```
cursor.execute('''
DELETE FROM dish
WHERE name = %s
''', [' 仰望星空 '])
conn.commit() #  插入、修改时务必
```

JSONB 内的增加，方法 1

```
SELECT
'{" 名称 ": " 仰望星空 "}'::jsonb
|| '{" 口味 ": {" 奇葩 ": " 满分 "}}'::jsonb

"{" 口味 ": {" 奇葩 ": " 满分 "}, " 名称 ": " 仰望星空 "}"
```

JSONB 内的增加，方法 2

```
SELECT
jsonb_set(
    '{" 名称 ": " 仰望星空 "}'::jsonb,
    '{ 口味 }',
    '{" 奇葩 ": " 满分 "}'::jsonb)
```

```
"{" 口味 ": {" 奇葩 ": " 满分 "}, " 名称 ": " 仰望星空 "}"
```

JSONB 的删除，方法 1

```
SELECT
'{" 人数 ":1," 名称 ": " 仰望星空 "}'::jsonb
- ' 名称 '
```

```
"{" 人数 ": 1}"
```

JSONB 的删除，方法 2

```
SELECT
'{" 口味 ":{" 奇葩 ":" 满分 "}}'::jsonb
#- '{ 口味 , 奇葩 }'


"{" 口味 ": {}}"
```

JSONB 的修改，方法 1

```
SELECT
'{" 口味 ":{" 奇葩 ":" 满分 "}}'::jsonb
|| '{" 口味 ":{" 惊艳 ":" 满分 "}}'::jsonb
```

```
"{" 口味 ": {" 惊艳 ": " 满分 "}}"
```

JSONB 的修改，方法 1

```
SELECT
jsonb_set(
    '{" 口味 ":{" 奇葩 ":" 满分 "}}'::jsonb,
    '{ 口味 , 奇葩 }',
    '" 完美 "'::jsonb)
```

```
"{" 口味 ": {" 奇葩 ": " 完美 "}}"
```

高级方法

- 复杂 JSON 的展开
- 使用 VIEW 构建复杂的视图
- 使用 FUNCTION 构建复杂的查询函数

复杂的 JSON


```
{
  "辅料": [
    {
      "名称": "牛奶",
      "用量": "100 克"
    },
    {
      "名称": "玉米淀粉",
      "用量": "40 克"
    }
  ]
}
```

展开 JSON

```
SELECT
jsonb_array_elements(
'{" 辅料 ":[{" 名称 ":" 牛奶 "," 用量 ":"100 克 "},{" 名称 ":" 玉米淀粉 "," 用量 ":"40 克 "}]}'::jsonb
-> ' 辅料 ')
```

```
"{" 名称 ": " 牛奶 ", " 用量 ": "100 克 "}"
"{" 名称 ": " 玉米淀粉 ", " 用量 ": "40 克 "}"
```

视图的优点

- 定制数据
- 简化操作
- 数据紧随表格
- 原始表安全

JSON 视图

```
CREATE MATERIALIZED VIEW dish_stuff AS
SELECT name, stuff, nullif(g, '')::bigint as g
FROM (
    SELECT
    name,
    jsonb_array_elements(stuff)->>'名称' AS stuff,
    jsonb_array_elements(stuff)->>'用量' AS detail,
    regexp_replace(jsonb_array_elements(stuff)->>'用量', '[^\d]+', '', 'g') AS g
    FROM (
        SELECT name, data->'辅料' AS stuff
        FROM dish
        WHERE data ? '辅料'
    ) temp1
) temp2;
```

SQL 函数

- 自由的查询参数
- 提高效率
- 原子操作，可回退性

复杂函数示例

```
CREATE OR REPLACE FUNCTION
find_dish_calories (dish_name text)
RETURNS TABLE
(name text, stuff text, calories numeric)
AS $$
    SELECT
        dish_stuff.name as name,
        string_agg(stuff, ',') as stuff,
        sum(g / 100.0 * calories) AS calories
    FROM dish_stuff JOIN stuff_calories
    ON dish_stuff.stuff = stuff_calories.name
    WHERE dish_stuff.name = dish_name
    GROUP BY dish_stuff.name
$$ LANGUAGE SQL;
```

使用函数

```
select * from find_dish_calories(' 华夫饼 ');

select * from find_dish_calories(' 南瓜饼 ');

select * from find_dish_calories(' 红烧排骨 ');
```

结束

谢谢！