Untitled Notebook

Anonymous

2025 年 6 月 20 日

```
[57]:
import sqlite3
import os



conn = sqlite3.connect('family.db')
cursor = conn.cursor()

cursor.execute('DROP TABLE IF EXISTS family;')
# Drop the family table if it already exists
cursor.execute('''
    CREATE TABLE family (
    father TEXT,
    son TEXT
);
''')
```

```
<sqlite3.Cursor at 0x1da01bf9940>
```

```
[58]:
# Insert the values
insert_data = [
    ('司马防', '司马懿'),
    ('司马防', '司马孚'),
    ('司马防', '司马馗'),
    ('司马懿', '司马师'),
    ('司马懿', '司马昭'),
    ('司马懿', '司马亮'),
    ('司马懿', '司马伦'),
    ('司马孚', '司马瑰'),
    ('司马馗', '司马泰'),
    ('司马师', '司马攸'),
    ('司马昭', '司马炎'),
    ('司马瑰', '司马颙'),
    ('司马攸', '司马囧'),
    ('司马炎', '司马衷'),
    ('司马炎', '司马玮'),
    ('司马炎', '司马乂'),
    ('司马炎', '司马颖'),
    ('司马炎', '司马炽')
]

cursor.executemany('INSERT INTO family VALUES (?, ?)', insert_data)
conn.commit()
```

```
#1. Brother rule: brother(X,Y) :- father(Z,X), father(Z,Y)
def find_brothers():
    cursor=conn.cursor()
    cursor.execute('''
    SELECT f1.son AS son1,f2.son AS son2
    FROM family f1
    JOIN family f2 on f1.father=f2.father
    WHERE son1 < son2
    ORDER BY son1,son2

    ''')
    brothers = cursor.fetchall()
    print("Brothers(无重复):")
    for b in brothers:
        print(f"({b[0]},{b[1]})")
```

[59]:

[60]:
```
# 2. Ancestor rules:
#    ancestor(X,Y) :- father(X,Y)
#    ancestor(X,Y) :- father(X,Z), ancestor(Z,Y)
def find_ancestors():
    # This requires a recursive query
    cursor = conn.cursor()
    cursor.execute('''
    WITH RECURSIVE ancestor(ancestor, descendant) AS (
        -- Base case: direct father-son relationships
        SELECT father, son FROM family

        UNION

        -- Recursive case: father of someone who is already an ancestor
        SELECT f.father, a.descendant
        FROM family f
        JOIN ancestor a ON f.son = a.ancestor
    )
    SELECT * FROM ancestor ORDER BY ancestor, descendant
    ''')
    ancestors = cursor.fetchall()
    print("\nAncestors:")
    print("左边是右边的祖先: ")
    for a in ancestors:
        print(f"({a[0]},{a[1]})")
```

[61]:
```
# Execute the queries
find_brothers()
find_ancestors()
```

```
Brothers(无重复):(司马乂,司马炽)(司马乂,司马玮)(司马乂,司马衷)(司马乂,司马颖)(司马
亮,司马伦)(司马亮,司马师)(司马亮,司马昭)(司马伦,司马师)(司马伦,司马昭)(司马孚,司马
懿)(司马孚,司马馗)(司马师,司马昭)(司马懿,司马馗)(司马炽,司马玮)(司马炽,司马衷)(司
马炽,司马颖)(司马玮,司马衷)(司马玮,司马颖)(司马衷,司马颖)Ancestors:左边是右边的祖
先:(司马孚,司马瑰)(司马孚,司马颙)(司马师,司马冏)(司马师,司马攸)(司马懿,司马乂)(司
马懿,司马亮)(司马懿,司马伦)(司马懿,司马冏)(司马懿,司马师)(司马懿,司马攸)(司马懿,
司马昭)(司马懿,司马炎)(司马懿,司马炽)(司马懿,司马玮)(司马懿,司马衷)(司马懿,司马颖)
(司马攸,司马冏)(司马昭,司马乂)(司马昭,司马炎)(司马昭,司马炽)(司马昭,司马玮)(司马
昭,司马衷)(司马昭,司马颖)(司马炎,司马乂)(司马炎,司马炽)(司马炎,司马玮)(司马炎,司马
衷)(司马炎,司马颖)(司马瑰,司马颙)(司马防,司马乂)(司马防,司马亮)(司马防,司马伦)(司
马防,司马冏)(司马防,司马孚)(司马防,司马师)(司马防,司马懿)(司马防,司马攸)(司马防,
司马昭)(司马防,司马泰)(司马防,司马炎)(司马防,司马炽)(司马防,司马玮)(司马防,司马瑰)
(司马防,司马衷)(司马防,司马颖)(司马防,司马颙)(司马防,司马馗)(司马馗,司马泰)
```

```python
#GROUP_CONCAT
def find_ancestors():
    cursor = conn.cursor()
    cursor.execute('''
    WITH RECURSIVE ancestor(ancestor, descendant) AS (
        -- 基础情况: 直接的父子关系
        SELECT father, son FROM family

        UNION

        -- 递归情况: 祖先的祖先也是祖先
        SELECT f.father, a.descendant
        FROM family f
        JOIN ancestor a ON f.son = a.ancestor
    )
    SELECT
        ancestor,
        GROUP_CONCAT(descendant, '、') AS descendants
    FROM ancestor
    GROUP BY ancestor
    ORDER BY ancestor
    ''')

    ancestors = cursor.fetchall()
    print("\n祖先及其后辈列表: ")
   print("=====================================================================================
    for ancestor, descendants in ancestors:
        print(f"{ancestor}: {descendants}")

# 调用函数
```

```python
[62]: find_ancestors()
```

```
祖先及其后辈列表:=====================================================================================
司马孚: 司马瑰、司马颙司马师: 司马攸、司马冏司马懿: 司马师、司马昭、司马亮、司马伦、
司马攸、司马炎、司马冏、司马衷、司马玮、司马乂、司马颖、司马炽司马攸: 司马冏司马昭:
司马炎、司马衷、司马玮、司马乂、司马颖、司马炽司马炎: 司马衷、司马玮、司马乂、司马颖、
司马炽司马瑰: 司马颙司马防: 司马懿、司马孚、司马馗、司马师、司马昭、司马亮、司马伦、
司马瑰、司马泰、司马攸、司马炎、司马颙、司马冏、司马衷、司马玮、司马乂、司马颖、司马
炽司马馗: 司马泰
```

```python
[63]: conn.close()
```