Untitled Notebook

Anonymous

2025 年 6 月 20 日

[1]:
```
import sqlite3
import json

# 创建数据库连接
conn = sqlite3.connect('database_course.db')
c = conn.cursor()

# 创建存储JSON的原始表
c.execute('''CREATE TABLE IF NOT EXISTS chapter_json (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                content JSON
             )''')
```

```
<sqlite3.Cursor at 0x175a50b0540>
```

```
# 示例JSON数据
data = {
    "chapter": "关系规范化理论",
    "sections": [
        {
            "section_title": "基本概念与问题引入",
            "knowledge_points": [
                {
                    "key_point": "关系模式设计问题",
                      "explanation": "非规范化关系模式会导致数据冗余、插入异常、更新异常和删除异常。例如学生选课表中存储系主任信息时，会导致系主任信息重复存储（冗余）且无法独立维护（异常）",
                    "qa": {
                      "question": "非规范化关系模式可能引发哪些问题？请举例说明。",
                        "answer": "数据冗余（如系主任信息重复存储）、插入异常（无法单独插入未选课学生的系信息）、更新异常（修改系主任需更新多条记录）、删除异常（删除最后一条学生记录会丢失系信息）"
                    }
                },
                {
                    "key_point": "函数依赖",
                      "explanation": "描述属性间逻辑关系的约束，包括完全依赖、部分依赖和传递依赖。如学号→系名是完全依赖，(学号,课程)→成绩是部分依赖，学号→系主任是传递依赖",
                    "qa": {
                         "question": "什么是传递函数依赖？请用学生-系-系主任的例子说明。",
                        "answer": "若存在学号→系名，系名→系主任，且系名↛学号，则系主任传递依赖于学号"
                    }
                }
            ]
        },
        {
```

```
            "section_title": "范式体系",
            "knowledge_points": [
                {
                    "key_point": "第一范式（1NF）",
                     "explanation": "属性值不可再分，消除重复组。如将包含多值的地址
字段拆分为省、市、街道",
                    "qa": {
                       "question": "判断表结构是否满足1NF：商品表(商品ID，商品名称，
规格['红色','L'])",
                         "answer": "不满足1NF，'规格'字段包含多个值，需拆分为独立属
性或新建规格表"
                    }
                },
                # 其他知识点...
            ]
        },
        # 其他章节...
    ],
    "question_bank": [
        {
            "question": "Armstrong公理包含哪些基本规则？",
             "answer": "自反律（若Y⊆X则X→Y）、增广律（X→Y则XZ→YZ）、传递律（X→Y且Y→Z
则X→Z）"
        },
        # 其他问题...
    ]
[2]: }
```

```
# 插入JSON数据
c.execute("INSERT INTO chapter_json (content) VALUES (?)",
          (json.dumps(data, ensure_ascii=False),))
[3]: conn.commit()
```

```
# 查询1：获取特定字段值（所有章节标题）
print("查询1：所有章节标题")
c.execute('''SELECT json_extract(content, '$.chapter')
             FROM chapter_json''')
[4]: print(c.fetchone()[0])
```

```
查询1：所有章节标题关系规范化理论
```

```
# 查询2：统计特定数组元素数目（每个section的知识点数量）
print("\n查询2：各章节知识点数量")
c.execute('''SELECT json_extract(s.value, '$.section_title'),
                    json_array_length(s.value, '$.knowledge_points')
             FROM chapter_json,
                  json_each(chapter_json.content, '$.sections') AS s''')
for row in c.fetchall():
[5]:     print(f"{row[0]}: {row[1]}个知识点")
```

```
查询2：各章节知识点数量基本概念与问题引入：2个知识点范式体系：1个知识点
```

```
# 创建规范化平面表（1NF）
c.execute('''CREATE TABLE IF NOT EXISTS normalized_chapter (
                chapter TEXT,
                section_title TEXT,
                key_point TEXT,
                explanation TEXT,
                question TEXT,
                answer TEXT
            )''')

# 展开JSON结构插入平面表
c.execute('''INSERT INTO normalized_chapter
            SELECT json_extract(content, '$.chapter'),
                   json_extract(s.value, '$.section_title'),
                   json_extract(kp.value, '$.key_point'),
                   json_extract(kp.value, '$.explanation'),
                   json_extract(kp.value, '$.qa.question'),
                   json_extract(kp.value, '$.qa.answer')
            FROM chapter_json,
                 json_each(chapter_json.content, '$.sections') AS s,
                 json_each(s.value, '$.knowledge_points') AS kp''')
[6]: conn.commit()
```

```
# 验证规范化表
print("\n规范化表数据示例:")
c.execute("SELECT * FROM normalized_chapter LIMIT 2")
for row in c.fetchall():
    print(row)

# 创建问题库表
c.execute('''CREATE TABLE IF NOT EXISTS question_bank (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                chapter TEXT,
                question TEXT,
                answer TEXT
            )''')

# 插入问题库数据
c.execute('''INSERT INTO question_bank (chapter, question, answer)
            SELECT json_extract(content, '$.chapter'),
                   json_extract(q.value, '$.question'),
                   json_extract(q.value, '$.answer')
            FROM chapter_json,
                 json_each(chapter_json.content, '$.question_bank') AS q''')
[7]: conn.commit()
```

```
规范化表数据示例:('关系规范化理论', '基本概念与问题引入', '关系模式设计问题', '非规范化关系模式会导致数据冗余、插入异常、更新异常和删除异常。例如学生选课表中存储系主任信息时，会导致系主任信息重复存储（冗余）且无法独立维护（异常）', '非规范化关系模式可能引发哪些问题？请举例说明。', '数据冗余（如系主任信息重复存储）、插入异常（无法单独插入未选课学生的系信息）、更新异常（修改系主任需更新多条记录）、删除异常（删除最后一条学生记录会丢失系信息）')('关系规范化理论', '基本概念与问题引入', '函数依赖', '描述属性间逻辑关系的约束，包括完全依赖、部分依赖和传递依赖。如学号→系名是完全依赖，(学号,课程)→成绩是部分依赖，学号→系主任是传递依赖', '什么是传递函数依赖？请用学生-系-系主任的例子说明。', '若存在学号→系名，系名→系主任，且系名↛学号，则系主任传递依赖于学号')
```