=== Replit 账单支持多银行格式、批量处理、自动分类规则解析系统架构升级指南 ===

致：Replit 开发团队 / INFINITE GZ 技术负责人
时间：2025年11月15日
主题：将OCR解析方案升级为结构化数据解析方案

==========
一、问题背景
==========

当前Replit系统使用OCR技术解析：
1. 信用卡账单
2. 个人/公司银行流水月结单

存在以下主要问题：

❌ 准确率不足：
- OCR识别率仅为70-85%
- 数字混淆：0→O、1→l、5→S
- 金额错误导致财务数据不准确

❌ 处理速度慢：
- 单份账单需要20-40秒
- PDF转图片 + OCR识别 + 后处理流程复杂

❌ 成本高：
- Google Vision API / AWS Textract 费用
- 每月1000份账单需$1.5-3.0

❌ 维护难度大：
- OCR模型黑盒，难调试
- 需频繁处理边界情况

==========
二、新方案：结构化数据解析
==========

✅ 核心优势：

1. 准确率 99%+
- 直接读取Excel/CSV原始数据
- 零数字识别错误
- 保留小数精度

2. 处理速度提升30倍
- 单份账单仅需 0.6-1秒
- 直接解析，无中间步骤

3. 零成本
  - 不需OCR API费用
  - Replit免费层足够支持

4. 易维护
  - 规则透明，可调试
  - Python代码完全可控

==========
三、需要替换的两大模块
==========

■ 模块A：信用卡账单解析器
——————————————————

当前状态：使用OCR识别PDF信用卡账单

需替换为：结构化数据解析

支持功能：
• 自动读取账单基本信息：
  - Owner's Name
  - Bank Name
  - Credit Card Last 4 digits
  - Card Type (Visa/Master/Amex)
  - Statement Date
  - Due Date
  - Card Limit
  - Previous Balance
  - Closing Balance

• 自动拆分DR/CR：
  - DR (Debit) = 支出/消费
  - CR (Credit) = 还款/退款

• 逐笔Running Balance计算

• 智能分类：
  - Purchases (消费)
  - Payment (还款)
  - Finance Charges (利息费用)
  - Instalment Details (BT/Flexi/Cash)

• 余额核对：
  自动验证 Closing Balance 一致性

■ 模块B：银行流水月结单解析器
——————————————————————

当前状态：使用OCR识别PDF银行账单

需替换为：结构化数据解析

支持银行：
• Public Bank
• Maybank
• CIMB
• RHB
• Hong Leong Bank
• AmBank
• Alliance Bank

支持功能：
• 自动读取账户信息：
  - Account Number
  - Account Type
  - Account Holder Name
  - Statement Date
  - Opening Balance
  - Closing Balance
  - Total Debits
  - Total Credits

• 自动拆分DR/CR：
  - DR (Debit) = 支出/转出
  - CR (Credit) = 收入/转入

• 逐笔Running Balance计算

• 智能30+类别分类：
  - 收入类：薪资、利息、股息、退款
  - 支出类：银行费用、转账、ATM提款
  - 账单类：水电费、通讯费、网购、外卖
  - 消费类：汽油费、保险、贷款还款
  - 其他：投资、教育、医疗、政府税费

• 批量处理：
  支持多份账单一次性处理

• 余额核对：
  自动验证 Closing Balance 一致性

==========
四、Replit具体实施指导
==========

步骤1：创建Replit项目结构
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

```
replit-statement-parser/
├── main.py                    # FastAPI主程序
├── parsers/
│   ├── __init__.py
│   ├── credit_card_parser.py  # 信用卡解析器
│   ├── bank_statement_parser.py # 银行账单解析器
│   ├── bank_detector.py       # 银行识别
│   └── classifier.py          # 智能分类
├── config/
│   ├── bank_templates.json    # 银行模板
│   └── classification_rules.json # 分类规则
├── requirements.txt
└── .replit
```

步骤2：API端点设计
━━━━━━━━━━━━━━━━━━━━━━━━━━

请在Replit中实现以下3个API端点：

1. POST /parse/credit-card
   功能：解析信用卡账单
   输入：Excel/CSV文件
   输出：JSON格式的结构化数据

2. POST /parse/bank-statement
   功能：解析银行流水月结单
   输入：Excel/CSV文件
   输出：JSON格式的结构化数据

3. POST /parse/batch
   功能：批量处理多份账单
   输入：多个Excel/CSV文件
   输出：批量处理结果

步骤3：输出JSON格式规范
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

【信用卡账单JSON示例】

```
{
  "status": "success",
  "document_type": "credit_card",
  "account_info": {
    "owner_name": "CHANG CHOON CHOW",
    "bank": "PUBLIC BANK",
```

```
    "card_last_4": "1234",
    "card_type": "Visa",
    "statement_date": "12-09-2024",
    "due_date": "02-10-2024",
    "card_limit": 10000.00,
    "previous_balance": 5000.00,
    "closing_balance": 3500.00
  },
  "transactions": [
    {
      "date": "01-09-2024",
      "posting_date": "01-09-2024",
      "description": "SHOPEE PAYMENT",
      "amount": 150.00,
      "dr": 150.00,
      "cr": 0,
      "running_balance": 5150.00,
      "category": "Purchases",
      "sub_category": "网购"
    },
    {
      "date": "05-09-2024",
      "posting_date": "05-09-2024",
      "description": "PAYMENT - THANK YOU",
      "amount": 2000.00,
      "dr": 0,
      "cr": 2000.00,
      "running_balance": 3150.00,
      "category": "Payment",
      "sub_category": "还款"
    }
  ],
  "summary": {
    "total_transactions": 25,
    "total_purchases": 4500.00,
    "total_payments": 6000.00,
    "total_finance_charges": 0,
    "balance_verified": true
  }
}
```


【银行流水月结单JSON示例】


```
{
  "status": "success",
  "document_type": "bank_statement",
  "account_info": {
    "account_number": "3119090727",
```

```
    "account_type": "RM ACE Account",
    "account_holder": "CHANG CHOON CHOW",
    "bank": "PUBLIC BANK",
    "statement_date": "25-09-2024",
    "opening_balance": 469.31,
    "closing_balance": 598.19,
    "total_debits": 44600.90,
    "total_credits": 44729.78
  },
  "transactions": [
    {
      "date": "01-09-2024",
      "description": "TNB BILL PAYMENT",
      "debit": 150.00,
      "credit": 0,
      "running_balance": 319.31,
      "category": "水电费",
      "sub_category": "公共事业"
    },
    {
      "date": "05-09-2024",
      "description": "SALARY CREDIT",
      "debit": 0,
      "credit": 5000.00,
      "running_balance": 5319.31,
      "category": "薪资收入",
      "sub_category": "工资"
    }
  ],
  "summary": {
    "total_transactions": 40,
    "category_breakdown": {
      "薪资收入": {"count": 2, "amount": 10000.00},
      "水电费": {"count": 3, "amount": 450.00},
      "通讯费": {"count": 2, "amount": 200.00},
      "网购": {"count": 8, "amount": 1250.00}
    },
    "balance_verified": true
  }
}
```


==========

五、代码交付清单

==========

请Replit开发团队完成以下交付物：

☑️ 核心代码文件：

1. main.py - FastAPI主服务
2. parsers/credit_card_parser.py
3. parsers/bank_statement_parser.py
4. parsers/bank_detector.py
5. parsers/classifier.py
6. config/bank_templates.json
7. config/classification_rules.json

☑️ 配置文件:
- requirements.txt
- .replit

☑️ API文档:
- Swagger UI (/docs)
- API说明文档

☑️ 测试用例:
- 信用卡账单样例文件
- 银行账单样例文件
- Postman/cURL测试命令

==========
六、集成到n8n工作流
==========

n8n工作流示例:

1. Webhook节点:接收上传的账单文件
↓
2. HTTP Request节点:调用Replit API
- URL: https://your-replit.repl.co/parse/credit-card
- Method: POST
- Body: multipart/form-data
↓
3. Code节点:处理返回JSON数据
↓
4. PostgreSQL节点:写入数据库
- 表1: accounts (账户信息)
- 表2: transactions (交易明细)
- 表3: category_summary (分类统计)
↓
5. Email节点:发送处理通知

==========
七、数据库Schema设计
==========

表1: accounts (账户信息表)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
- id (UUID, Primary Key)
- document_type (VARCHAR) # credit_card / bank_statement
- owner_name (VARCHAR)
- bank (VARCHAR)
- account_number (VARCHAR)
- statement_date (DATE)
- opening_balance (DECIMAL)
- closing_balance (DECIMAL)
- total_debits (DECIMAL)
- total_credits (DECIMAL)
- balance_verified (BOOLEAN)
- created_at (TIMESTAMP)
- updated_at (TIMESTAMP)

表2: transactions (交易明细表)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
- id (UUID, Primary Key)
- account_id (UUID, Foreign Key)
- transaction_date (DATE)
- posting_date (DATE)
- description (TEXT)
- amount (DECIMAL)
- debit (DECIMAL)
- credit (DECIMAL)
- running_balance (DECIMAL)
- category (VARCHAR)
- sub_category (VARCHAR)
- created_at (TIMESTAMP)

表3: category_summary (分类统计表)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
- id (UUID, Primary Key)
- account_id (UUID, Foreign Key)
- category (VARCHAR)
- transaction_count (INTEGER)
- total_amount (DECIMAL)
- percentage (DECIMAL)
- period (VARCHAR) # YYYY-MM
- created_at (TIMESTAMP)

==========
八、实施时间表与验收标准
==========

阶段一：开发阶段 (5-7工作日)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
• 完成核心代码开发

• 实现两大模块解析器
• 配置类分类规则
• 本地开发环境测试

阶段二：Replit部署 (1-2工作日)
──────────────────────────────────────
• 上传代码到Replit
• 配置环境变量
• 启动API服务
• 生成API文档

阶段三：集成测试 (2-3工作日)
──────────────────────────────────────
• 配置n8n工作流
• 测试完整数据流转
• 数据库写入验证
• 边界情况测试

验收标准：
──────────
✅ 信用卡账单解析准确率 > 98%
✅ 银行账单解析准确率 > 98%
✅ 单份账单处理时间 < 2秒
✅ 余额核对100%通过
✅ 智能分类覆盖率 > 90%
✅ API响应时间 < 3秒
✅ 批量处理无失败

==========
九、技术支持联系人
==========

项目负责人：[Your Name]
公司：INFINITE GZ SDN BHD
联系方式：[Your Email/Phone]

技术问题反馈：
• 如遇Replit部署问题，请及时沟通
• 如需追加新银行支持，提供样例文件
• 分类规则需调整，提供具体需求

项目文档位置：
• VBA代码：[GitHub Gist Link]
• Python代码示例：本文档附件
• n8n工作流配置：待Replit完成后提供

==========

✅ 总结：

本文档已详细说明：
1. 为什么要从CR升级到结构化解析
2. 两大模块的具体功能要求
3. API设计和JSON输出格式
4. n8n集成和数据库Schema
5. 实施计划和验收标准

请Replit团队按照此文档，将现有OCR方案
完全替换为新的结构化数据解析方案。

如有任何疑问，请立即反馈！

==========