جامعة دمشق – كلية الهندسة المعلوماتية

السنة الرابعة - قسم الذكاء الصنعي

عملي مترجمات

مشروع المترجمات

المرحلة الثانية Syntax Analysis:

تهدف هذه المرحلة إلى التحقق من أن سلسلة من المفردات (Tokens) التي قام المحلل المفرداتي بتمييزيها متوافقة مع القواعد (Grammar) الخاص بلغة البرمجة في حالتنا (SQL) وبالتالي فهي تجيب على السؤال هل تنتمي سلسلة المفردات الموجودة في ملف تعليمات الدخل إلى اللغة التي يولدها الGrammar ؟

المطلوب

1- كتابة مجموعة القواعد ال Grammar للغة البرمجة SQL والتي تحدد شكل وبنية التعليمات البرمجية الصحيحة التي يمكن كتابتها في لغة البرمجة SQL باستخدام الاداة Antlr4/parser وذلك بوضع القواعد في ملف Parser Grammar

يجب أن تسمح القواعد المكتوبة بالتعرف على التعليمات الأساسية بلغة البرمجة SQL .

- DML Statements
  - select_statement
  - update_statement
  - delete_statement
  - insert_statement
  - .....
- DDL Statements
  - DROP
  - ALTER
  - CREATE
  - ....
- Cursor Manipulation Statements
- Common Table Expression Statemen (CTE)

2- حل التضارب في القواعد من خلال استخدام مفاهيم Precedence and Associativity للعمليات الحسابية والمنطقية و ...

الاعتماد على **Transact-SQL syntax documentation** كمرجع مساعد في كتابة القواعد

الرابط https://learn.microsoft.com/en-us/sql/t-sql/queries/queries?view=sql-server-ver17

نختار مثلا من قائمة التعليمات تعليمة UPDATE (Transact-SQL) نجد في الصفحة syntax التعليمة

```
-- Syntax for SQL Server and Azure SQL Database
[ WITH <common_table_expression> [...n] ]
UPDATE
    [ TOP ( expression ) [ PERCENT ] ]
    { { table_alias | <object> | rowset_function_limited
         [ WITH ( <Table_Hint_Limited> [ ...n ] ) ]
      }
      | @table_variable
    }
    SET
        { column_name = { expression | DEFAULT | NULL }
          | { udt_column_name.{ { property_name = expression
                                | field_name = expression }
                                | method_name ( argument [ ,...n ] )
                              }
          }
          | column_name { .WRITE ( expression , @Offset , @Length ) }
          | @variable = expression
          | @variable = column = expression
          | column_name { += | -= | *= | /= | %= | &= | ^= | |= } expression
          | @variable { += | -= | *= | /= | %= | &= | ^= | |= } expression
          | @variable = column { += | -= | *= | /= | %= | &= | ^= | |= } expression
        } [ ,...n ]

    [ <OUTPUT Clause> ]
    [ FROM{ <table_source> } [ ,...n ] ]
    [ WHERE { <search_condition>
            | { [ CURRENT OF
                  { { [ GLOBAL ] cursor_name }
                      | cursor_variable_name
                  }
                ]
              }
            }
    ]
    [ OPTION ( <query_hint> [ ,...n ] ) ]
[ ; ]

<object> ::=
{
    [ server_name . database_name . schema_name .
    | database_name .[ schema_name ] .
    | schema_name .
    ]
    table_or_view_name}
```

كيف نقرأ هذا ال syntax ونفهمه فالترميز المستخدم (notation) في SQL documentation يختلف عن الترميز الذي سنكتب به في ملف القواعد لل antlr4 مايلي شرح للترميز المستخدم ومايقابله في ال antlr4

1. Angle brackets <...>

في كل قاعدة نكتبها : الطرف اليميني للقاعدة من الممكن ان يحوي

- رموز terminal (Tokens)
- non terminal (grammar rule) معرفة في مكان اخر

الترميز <...> يعني Non-terminal placeholder مثال <common_table_expression> هذا يعني ان common_table_expression هي قاعدة وهذه القاعدة موجودة (في قسم arguments بالاسفل تجدها)

2. Square brackets [. . .]

مثال [WHERE <search_condition>] تعني ان التعليمة قد تأتي مرة او لا تأتي ابدا

مايقابلها في ترميز ال antlr4

whereClause?

3. Curly braces { ... }

مثال

{ table_alias | <object> | rowset_function_limited }

تعني اختيار واحد مايقابله في antlr4

tableAlias
| objectName
| rowsetFunctionLimited

| Vertical bar نفس المعنى في الantlr4

4. Ellipsis [,...n]

Repeat zero or more times (comma-separated)

[,...n] SET assignment وكمثال في ال sql SET a=1, b=2, c=3

اذن [,...n] assignment يقابله في antlr

assignment (',' assignment)*

5. Keywords in UPPERCASE (Tokens)

تعني الرموز التي ارسلها الlexer

ملاحظة هناك رموز ليست TOKEN وليست معرفة ضمن <> هذه ايضا يجب ان تكون قواعد معرفة في ملف قواعد المحلل النحوي ولكن الفرق مثلا بين قاعدتي <common_table_expression> و column_name ان common_table_expression معرفة بشكل رسمي ضمن ال DOCUMENTAIONاما column_name ليس لها تعريف رسمي فيمكن كتابة القاعدة لها مثلا على الشكل

columnName :

IDENTIFIER;

ملاحظات :

- استخدم وثائق اللغة (Documentation) كمرجع لفهم تعليمات اللغة وبنيتها العامة،
  ولا تحاول عكس الصياغة المكتوبة في الـ Syntax Documentation كما هي إلى قواعدGrammar ،
  لأن الغرض من وثائق الـSyntax هو الشرح والفهم البشري، وليس الوصف الشكلي الرسمي للغة.

- ابدأ دائمًا بمجموعة قواعد مصغّرة (Subset) لعدد محدود من التعليمات،
  ثم أضف الميزات بشكل تدريجي مع اختبار القواعد في كل مرحلة.
  على سبيل المثال، عند كتابة قواعد تعليمة:SELECT في المرحلة الأولى،
  اكتب قاعدة بسيطة تقبل الصيغة الأساسية فقط:

```
select column1, column2 from table1
```

بعد التأكد من صحة القاعدة واختبارها، انتقل إلى المرحلة الثانية وأضف جزء الشرط:

```
select column1, column2 from table1 where column1 = 'hello world'
```

بعد ذلك، يمكن توسيع القواعد تدريجيًا لإضافة:

- JOIN
- GROUP BY
- ORDER BY
- وغيرها من الميزات

هذا الأسلوب يقلل التعقيد، ويساعد على بناء Grammar صحيحة وقابلة للتوسّع.