MANUAL TÉCNICO

Requisitos:

- Cualquier computadora con windows o linux.
- Instalar python versión 3.6 o superior
- Instalar librerías con pip: ply, graphviz,prettytable.
- Instalar Pycharm o Visual Studio Code

Funcionalidad

PLY y GRAMÁTICA:

La gramática ascendente definida BNF.md se implemento en ply, la clase gammar2.py se compone de 2 apartados importantes. El primero el **análisis léxico**, que se compone de palabras reservadas que se enlistan en un arreglo y de algunas funciones que trabajan con expresiones regulares son capaces de retornar tokens.

Ejemplo para definir un identificador con una expresión regular :

```
def t_ID(t):
    r'[a-zA-Z_][a-zA-Z_0-9]*'
    t.type = reservadas.get(t.value.lower(), 'ID')
    return t
```

La otra parte se compone de el análisis sintáctico, aquí se implementan las reglas (acorde a la construcción de un AST enfocado a una base de datos) y las producciones acorde a la gramática:

Ejemplo de una condición:

```
def p_cond(p):
    """
    cond    :   id MAYOR valortipo
            |   id MENOR valortipo
            |   id IGUAL valortipo
            |   id MENOR_IGUAL valortipo
            |   id MAYOR_IGUAL valortipo
    """
    p[0] = inst.cond(p[1],p[2],p[3])
    insertProduction(p.slice, len(p.slice))
```

También se destacan una dos producciones de error en donde cae, valga la redundancia cualquier error según la documentación en esta ocasión utilizamos utilizamos un método alterno para la recuperación de errores, según la documentación de ply:

“This function simply discards the bad token and tells the parser that the error was ok.”

Lo que se traduce en que se descarta el error y que el token está bien, el código se recupera en la siguiente producción válida.

```
def p_error(t):
    if t:
        descript = 'error sintactico en el token ' + str(t.type)
        linea = str(t.lineno)
        columna = str(find_column(t))
        nuevo_error = CError(linea,columna,descript,'Sintactico')
        insert_error(nuevo_error)
        parser.errok()
    else:
        print("Syntax error at EOF")
    return
```

CLASES ABSTRACTAS

Las diferentes clases abstractas moldean las propiedades y atributos de una base de datos, cada clase cuenta con su método abstracto y todas las clases son capaces de invocar siempre y cuando se herede de la clase principal.

Ejemplo de una clase abstracta:

```
class select(query):

    def __init__(self, distinct=False, select_list=[], table_expression=[], condition=[], group=False, ha
        self.distinct = distinct
        self.select_list = select_list
        self.table_expression = table_expression
        self.condition = condition
        self.group = group
        self.having = having
        self.orderby = orderby
        self.limit = limit
        self.offset = offset
        if having is not None and condition is not None:
            self.condition.append(having)
```

SALIDA Y REPORTES

Se utilizaron dos librerías más para la parte de visualización de datos de datos, en que caso de graphviz es sencillo construir tablas y mostrarla en pdf, en el caso del reporte se usaron conceptos más avanzados, debido a que la gramática cambio conforme al desarrollo, se implementó una clase capaz de graficar cualquier otra clase abstracta, para ello en siguiente link explican como es que los objetos json se pueden transformar en un árbol. Python create tree from a JSON file (visbud.blogspot.com) la solución 1 es sencilla, utiliza recursión y joins, a partir de esta solución se saca un archivo de texto que es mucho más comprensible (no es un árbol en forma ast pero es un árbol que funciona con tabulaciones) por suerte para graficar ese archivo de texto nos encontramos con la siguiente solución algorithm - Python file parsing: Build tree from text file - Stack Overflow.

En el caso del pretty table es solo una librería que permite transformar arreglos a texto tabulado en una tabla, esto casa bien imprimir tablas la consola.

LICENCIAS:

Las 3 librerías utilizadas son compatibles con la licencia MIT por lo que no supone un problema el uso de estas librerías.