

4. SYSTEM FEATURES

4.1. GENERAL

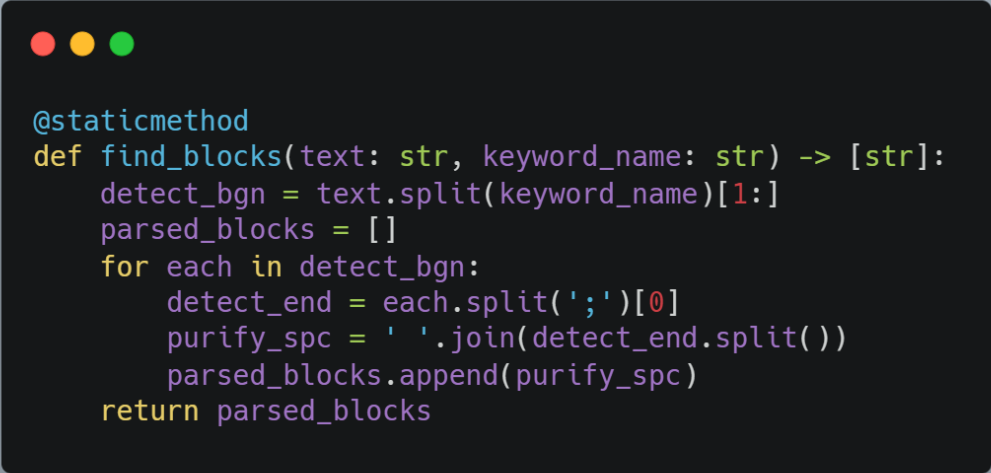
Since the project does not have an interface that interacts with the user, important functions of the project will be explained in detail in this section.

First of all, general stages will be mentioned. Then detailed descriptions of important functions will be mentioned. The project is based on three important phases.

- "Create Table" blocks in the sql script should be determined.
- The constraints of each column in each table should be known.
- For each table of old sql script, constraints differences between new sql script should be checked.

4.2. FIND CREATE TABLE BLOCK

The keyword "create table" is given to the function as input. The find_blocks function searches the given keyword in the script and determines the block from the point found to the first semicolon seen. It also cleans extra white spaces and prevents possible parse errors for next stage. Keeps how many blocks there are in the list.



```
@staticmethod
def find_blocks(text: str, keyword_name: str) -> [str]:
    detect_bgn = text.split(keyword_name)[1:]
    parsed_blocks = []
    for each in detect_bgn:
        detect_end = each.split(';')[0]
        purify_spc = ' '.join(detect_end.split())
        parsed_blocks.append(purify_spc)
    return parsed_blocks
```

4.3. TABLE PARSER

Finds the name of the table from begin to the first space character seen. In the remaining part (body), the pair of column name, column type and constraint separated by commas was parsed. For purifying the parsing process, if character is alphabetic or space it is accepted. From the remaining string, the column constraint is string that is found by searching in the constraints list. The column name is string that is not in both the constraints and the data type list.

```
def __table_parser(self, block: str) -> Table:
    name_idx = block.find(' ')
    name_str = block[:name_idx]
    body_idx = block.find('(')
    body_str = block[body_idx+1:-1]
    body = {}
    for line in body_str.split(','):
        col_part = ''.join(s for s in line if s.isalpha() or s.isspace()).split()
        col_name = ' '.join([s for s in col_part
                             if s not in self.__constraints and s not in self.__data_types])
        col_cons = [' '.join([s for s in col_part if s in self.__constraints])]
        if col_name in body:
            body[col_name] += col_cons
        else:
            body[col_name] = col_cons
    return Table(name_str, body)
```

4.4. CHECK DIFFERENCE

When looking for the differences between two sql scripts, it checks that the old table names should be a subset of the new table names and checks whether the columns that satisfy this condition have the same constraints. The errors that occur during these processes produce output to the log screen. The operations of adding a new table and adding a new column to an existing table are accepted.

```
def __check_diff(self):
    for new in self.__table_new:
        try:
            old = next(old for old in self.__table_old if old.name == new.name)
            if set(old.body).issubset(set(new.body)):
                for new_col_name, old_col_name in zip(new.body, old.body):
                    if new_col_name == old_col_name:
                        if set(new.body[new_col_name]) != set(old.body[old_col_name]):
                            tbl = 'TABLE NAME: {} | '.format(new.name)
                            col = 'COLUMN NAME: {} | '.format(new_col_name)
                            dif = 'DIFFERENCE: {} != {}'.format(new.body[new_col_name],
                                                                old.body[old_col_name])
                            self.__check_err.append(tbl + col + dif)
                    else:
                        self.__check_err.append('COLUMN REMOVE IS DETECTED IN \'{}\''
                                                .format(new.name))
            except StopIteration:
                self.__check_new.append(new)
    return False if self.__check_err else True
```