느낀 점들

<ALPHABET>과 <LEGAL IDENTIFIER> 순서. -> 한 글자인 경우.

마찬가지로 <DIGIT>와 <INT\_VALUE>의 순서. -> 한 글자.

KEYWORD 토큰들과 <LEGAL IDENTIFIER> 간의 순

한 글자 특수문자들을 < NON\_QUOTE\_SPECIAL\_CHARACTERS> 보다 먼저 정의해야 함.

<SPACE = “ “>와 같이 토큰을 정의하면 SKIP과 충돌이 일어나서, 그냥 “ “로 해줌.

KEYWORD 토큰에서 2단어로 이루어진 토큰(ex: CREATE TABLE)을 한 글자로 이루어진 2개의 토큰으로 분리함. 이러한 경우에서 사용자가 첫 단어를 입력한 뒤 Enter를 입력할 경우가 있다고 생각.

EBNF 문법을 JavaCC 문법에 맞춰 옮겨 적는 작업이 대부분의 시간을 소요했기 때문에 크게 난해한 부분은 없었음. 컴파일러 수업 때 while과 for문을 이용해 파싱했던 것을 생각하면 편한 라이브러리인 것 같다.

<COLUMN IN TABLE> ::= [<TABLE NAME> <PERIDO>] <COLUMN NAME>로 정의함. 자주 나와서 재사용하기 위해.

여기서 <TABLE NAME>과 <COLUMN NAME>이 정의상 모두 <LEGAL\_IDENTIFIER> 토큰이므로 2번의 LOOKAHEAD는 필수. <LEGAL IDENTIFIER>로 고치면 LOOKAHEAD가 필요 없어지지만, 따로 정의를 둔 이유가 있을 것 같아 고치지 않음.

<PREDICATE>은 4번의 LOOKAHEAD가 필요한데 지나치게 많음. 따라서 문법을 다음과 같이 수정해 2번의 LOOKAHEAD만 발생하도록 고침.

<PREDICATE> ::= <COMPARISON PREDICATE> | <NULL PREDICATE>

<COMPARISON PREDICATE> ::= <COMP OPERAND> <COMP OP> <COMP OPERAND>

<COMP OPERAND > ::= <COMPARABLE VALUE> | [<TABLE NAME> <PERIOD>] <COLUMN NAME>

<NULL PREDICATE> ::= [<TABLE NAME> <PERIDO>] <COLUMN NAME> <NULL OPERATION>

여기서 <COLUMN IN TABLE> ::= [<TABLE NAME> <PERIDO>] <COLUMN NAME>로 정의했으므로

<COMP OPERAND> ::= <COMPARABLE VALUE> | <COLUMN IN TABLE>

<NULL PREDICATE> ::= <COLUMN IN TABLE> <NULL OPERATION>

문제가 되는 <COMPARISON PREDICATE>의 or을 풀어 써서 <PREDICATE>에 대입하면

<PREDICATE> ::= <COLUMN IN TABLE> (<COMP OP> <COMP OPERAND> | <NULL OPERATION>) | <COMPARABLE VALUE> <COMP OP> <COMP OPERAND>

이 되는데, 큰 or에 묶이는 구문들을 따로 정의해주면, 최종적으로

<PREDICATE> ::= <COLUMN IN TABLE PREDICATE> | <COMPARABLE VALUE PREDICATE>

<COLUMN IN TABLE PREDICATE> ::= <COLUMN IN TABLE>

((<COMP OP> <COMP OPERAND>) | <NULL OPERATION>)

<COMPARABLE VALUE PREDICATE> ::= <COMPARABLE VALUE> <COMP OP> <COMP OPERAND>

와 같이 나타낼 수 있고, 해당 구문에서 발생하는 4회의 LOOKAHEAD를 <COLUMN IN TABLE>에서만 발생하는 2회로 줄일 수 있다.

뼈대 코드의 프린팅 방식이 비효율적인 것 같아 수정.

Option의 DEBUG\_PARSER는 문서를 찾아보니 default가 false가 불필요한 것 같아 삭제.

QA 7번은 틀린 토큰을 입력하고 enter를 누르면 바로 에러 메시지를 띄우는 것으로 함. Enter를 입력한 순간 이전 줄의 입력 값은 바꿀 수가 없으므로 에러가 입력되었다면 유저가 다음 줄에 무엇을 적든 에러. 따라서 빠르게 에러 처리되었다는 값을 받는 게 사용자 경험에 좋을 것 같아 에러 처리.

Ctrl+Z 누르면 에러가 무한 생성. <EOF> 처리 해줌.

Eclipse 환경이 불편해 JetBrain으로 환경 설정을 진행했으나 JetBrain의 Extension은 .jj파일로 .java파일 생성하는 기능을 지원하지 않아 Eclipse에서 진행. Git 사용은 불편해 CLI 사용.