CSCE 315

TEACHING ASSISTANT: JAY CHEN  JAYCHEN@CSE.TAMU.EDU
Project 1: **Database Management System**

- Design Document
- DB Engine
- **Lexical Parser**
- DB Engine and Lexical Parser Integration
- DB Application and integration
Baccus Naur Form (BNF):

- `=` : definition
- `,` : concatenation
- `;` : termination
- `|` : alternation, or
- `::=` : the symbol on the left must be replaced with the expression on the right
- `[ ]` : Optional items enclosed in square brackets
- `{}` : Items repeating 0 or more times
Grammar

insertcmd ::= 

INSERT INTO relationname VALUES FROM ( literal { , literal } );  \textbf{OR}  

INSERT INTO relationname VALUES FROM RELATION expr ;

\textbf{E.g.},  

INSERT INTO relationname VALUES FROM ( literal { , literal } );

\begin{tabular}{ | l | l | l |}
\hline
\textbf{animals} & \textbf{Dog} & \textbf{Cat} \\
\hline
Spot & Dog & 10 \\
Snoopy & Dog & 3 \\
Tweety & Bird & 1 \\
Joe & Cat & 4 \\
\hline
\end{tabular}
Grammar

E.g.,

`INSERT INTO relationname VALUES FROM RELATION expr`

`INSERT INTO species VALUES FROM RELATION project (kind) animals;`

<table>
<thead>
<tr>
<th>species</th>
<th>animals</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pig</td>
<td>Spot</td>
<td>Dog 10</td>
</tr>
<tr>
<td>dog</td>
<td>Snoopy</td>
<td>Dog 3</td>
</tr>
<tr>
<td>fish</td>
<td>Tweety</td>
<td>Bird 1</td>
</tr>
<tr>
<td></td>
<td>Joe</td>
<td>Cat 4</td>
</tr>
</tbody>
</table>

Note: both tables should have a “Primary Key” attribute, but it is ignored here
Grammar

\[
expr ::= \text{atomicexpr} \\
| \text{selection} \\
| \text{projection} \\
| \text{renaming} \\
| \text{union} \\
| \text{difference} \\
| \text{product}
\]

\[
selection ::= \text{select ( condition ) atomicexpr}
\]

\[
\text{atomicexpr ::= relationname | ( expr )}
\]

\[
\text{condition ::= conjunction \{ || conjunction \}}
\]

\[
\text{conjunction ::= comparison \{ && comparison \}}
\]

\[
\text{comparison ::= operand op operand | ( condition )}
\]

\[
\text{operand ::= attributename | literal}
\]

\[
\text{projection ::= project ( attributelist) atomicexpr}
\]

### A example expr:
Select (kind == “dog” || kind == “cat”) animal ;

Select (kind == “dog” || kind == “cat”) (project (kind, years) animal );

### animals

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spot</td>
<td>Dog</td>
<td>10</td>
</tr>
<tr>
<td>Snoopy</td>
<td>Dog</td>
<td>3</td>
</tr>
<tr>
<td>Tweety</td>
<td>Bird</td>
<td>1</td>
</tr>
<tr>
<td>Joe</td>
<td>Cat</td>
<td>4</td>
</tr>
</tbody>
</table>
Grammar

<table>
<thead>
<tr>
<th>Spot</th>
<th>Dog</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snoopy</td>
<td>Dog</td>
<td>3</td>
</tr>
<tr>
<td>Joe</td>
<td>Cat</td>
<td>4</td>
</tr>
</tbody>
</table>

Select (kind == “dog” || kind == “cat”) animal;

Select (kind == “dog” || kind == “cat”) (project (kind, years) animal);

Continue the previous example, but more complicated command:
Insert into species values from relation project (kind) animals;

Insert into species values from relation Select (kind == “dog” || kind == “cat”) (project (kind, years) animal);
Recursive Descent Parser

Grammar:
\[ E ::= x + T \]
\[ T ::= (E) \mid x \]

Example:
\[ x + (x + x) \]

- Entire \( x + (x + x) \) is an Expression
- \((x + x)\) is a Token
- \(x + x\) is an Expression
Recursive Descent Parser

Grammar:
\[ \text{selection} ::= \text{select ( condition ) atomicexpr} \]

Example:
Select (kind == “dog” || kind == “cat”) animal ;
Recursive Descent Parser

Select (kind == “dog” || kind == “cat”) animal;


```
Select (kind == “dog” || kind == “cat”) animal;
```

```
```
References

• https://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_Form
• https://en.wikipedia.org/wiki/Recursive_descent_parser
• http://www.cs.engr.uky.edu/~lewis/essays/compilers/rec-des.html