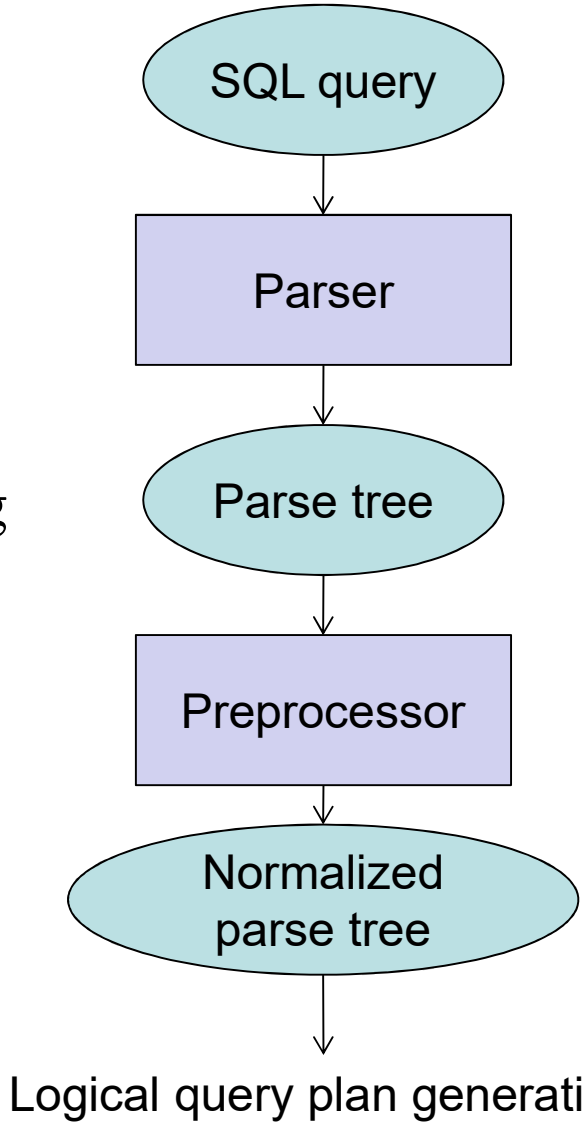
Database system implementation

# 2. Query parsing

# Query parsing outline

## Query parsing stages

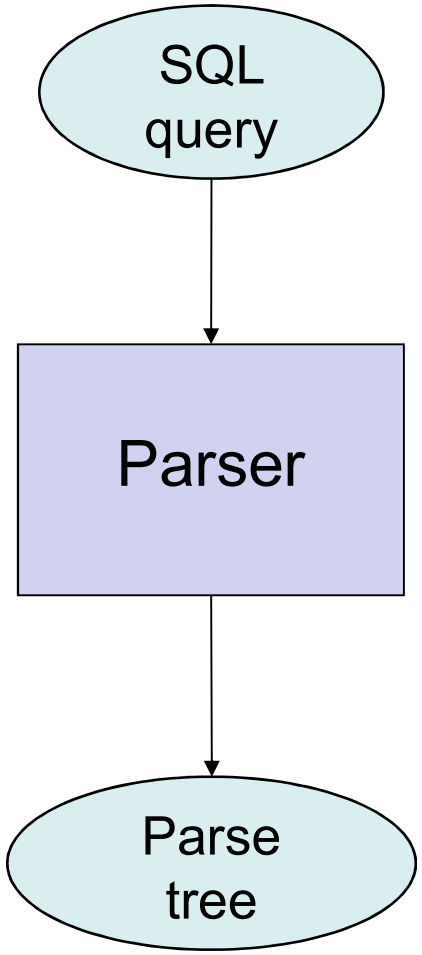
1. Syntax analysis (performed by *Parser*)
2. View resolution and Semantic checking (performed by *Preprocessor*)



# Syntax analysis

Parse tree consists of the nodes of the following two types:

1. ***Atoms*** - lexical elements such as keywords (for example, SELECT), names of attributes or relations, constants, parentheses, operators such as + or >, and others.
- ***Syntactic categories*** - which are names for families of query subparts. We shall represent syntactic categories by triangular brackets. For example, <SFW>, <Query>.



# A Grammar for a Simple Subset of SQL

**<Query>** ::= <SFW>
**<Query>** ::= ( <Query> )
<SFW> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>
<SelList> ::= <Attribute>, <SelList>
<SelList> ::= <Attribute>
<FromList> ::= <Relation>, <FromList>
<FromList> ::= <Relation>
<Condition> ::= <Condition> AND <Condition>
<Condition> ::= <Condition> OR <Condition>
<Condition> ::= NOT <Condition>
<Condition> ::= <Tuple> IN <Query>
<Condition> ::= <Attribute> = <Attribute>
<Condition> ::= <Attribute> < <Attribute>
<Condition> ::= <Attribute> > <Attribute>
. . . . . .
<Condition> ::= <Attribute> LIKE <Pattern>
<Condition> ::= <Attribute> EXIST <Pattern>
<Tuple> ::= <Attribute>

# Base Syntactic Categories

- <Attribute> = *identifier*
- <Relation> = *identifier*
- <Pattern> = *string*

# Example of query parsing

Names of suppliers, which supply at least one part in quantity more than 500 pieces.

```
SELECT   Name_S
FROM  S
WHERE  ID_S IN (
   /* IDs of suppliers, which supplier at least one part  in quantity more than 500 pieces */
   SELECT  ID_S
   FROM  SP
   WHERE  Quantity > 500
);
```

# Building the parse tree

<Query>

# SQL Grammar

**<Query> ::= <SFW>**

**<Query> ::= ( <Query> )**

<SFW> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>

<SelList> ::= <Attribute>, <SelList>

<SelList> ::= <Attribute>

<FromList> ::= <Relation>, <FromList>

<FromList> ::= <Relation>

<Condition> ::= <Condition> AND <Condition>

<Condition> ::= <Condition> OR <Condition>

<Condition> ::= NOT <Condition>

<Condition> ::= <Tuple> IN <Query>

<Condition> ::= <Attribute> = <Attribute>

<Condition> ::= <Attribute> < <Attribute>

<Condition> ::= <Attribute> > <Attribute>

…

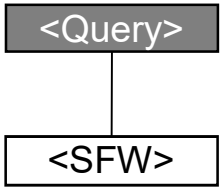
<Condition> ::= <Attribute> LIKE <Pattern>

<Condition> ::= <Attribute> EXIST <Pattern>

<Tuple> ::= <Attribute>

```
SELECT Name_S
FROM S
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500)
```

# Building the parse tree



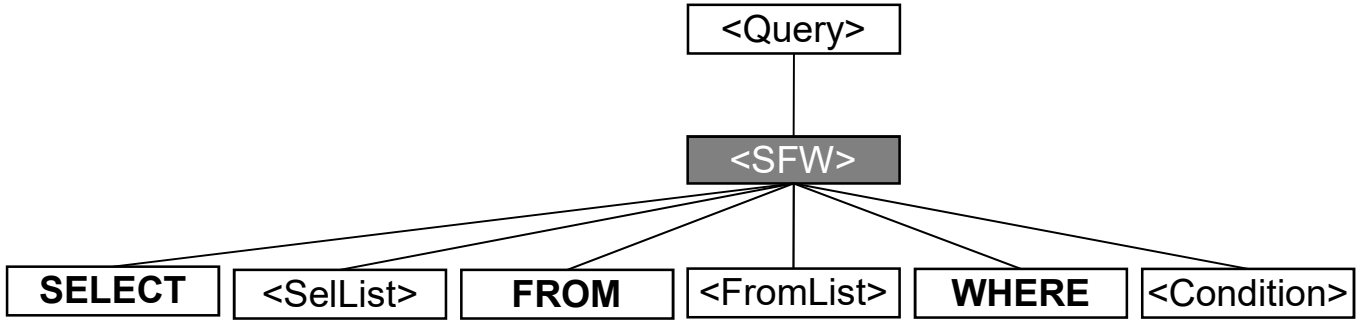

<Query> ::= <SFW>

# SQL Grammar

<Query> ::= <SFW>
<Query> ::= ( <Query> )
**<SFW> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>**
<SelList> ::= <Attribute>, <SelList>
<SelList> ::= <Attribute>
<FromList> ::= <Relation>, <FromList>
<FromList> ::= <Relation>
<Condition> ::= <Condition> AND <Condition>
<Condition> ::= <Condition> OR <Condition>
<Condition> ::= NOT <Condition>
<Condition> ::= <Tuple> IN <Query>
<Condition> ::= <Attribute> = <Attribute>
<Condition> ::= <Attribute> < <Attribute>
<Condition> ::= <Attribute> > <Attribute>
…
<Condition> ::= <Attribute> LIKE <Pattern>
<Condition> ::= <Attribute> EXIST <Pattern>
<Tuple> ::= <Attribute>

```
SELECT ID_S
FROM S
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500)
```

# Building the parse tree



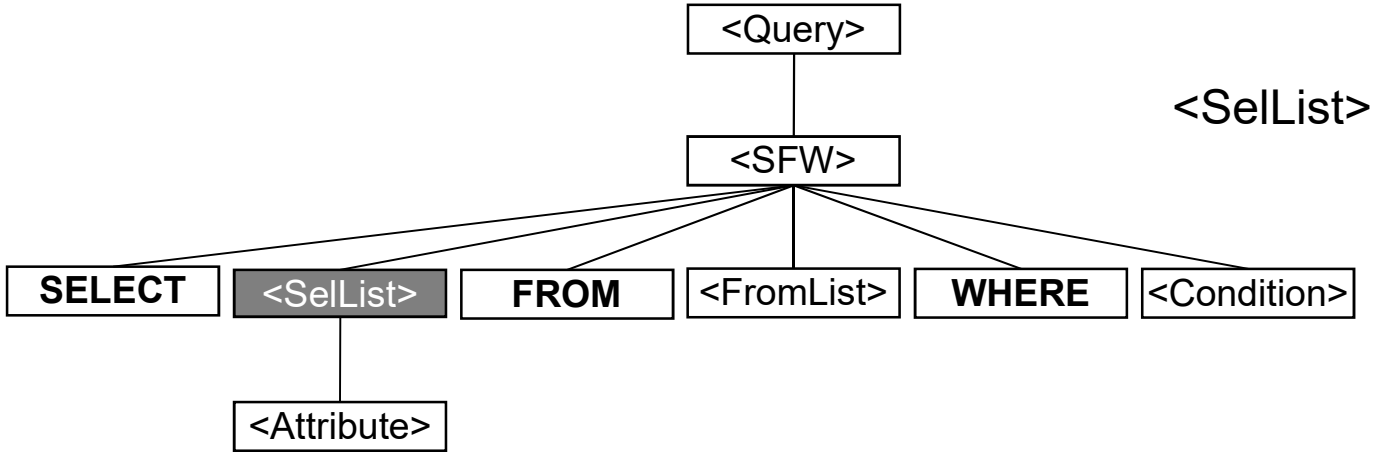

<SFW> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>

# SQL Grammar

<Query> ::= <SFW>
<Query> ::= ( <Query> )
<SFW> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>
**<SelList> ::= <Attribute>, <SelList>**
**<SelList> ::= <Attribute>**
<FromList> ::= <Relation>, <FromList>
<FromList> ::= <Relation>
<Condition> ::= <Condition> AND <Condition>
<Condition> ::= <Condition> OR <Condition>
<Condition> ::= NOT <Condition>
<Condition> ::= <Tuple> IN <Query>
<Condition> ::= <Attribute> = <Attribute>
<Condition> ::= <Attribute> < <Attribute>
<Condition> ::= <Attribute> > <Attribute>
…
<Condition> ::= <Attribute> LIKE <Pattern>
<Condition> ::= <Attribute> EXIST <Pattern>
<Tuple> ::= <Attribute>

```
SELECT Name_S
FROM S
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500);
```

# Building the parse tree



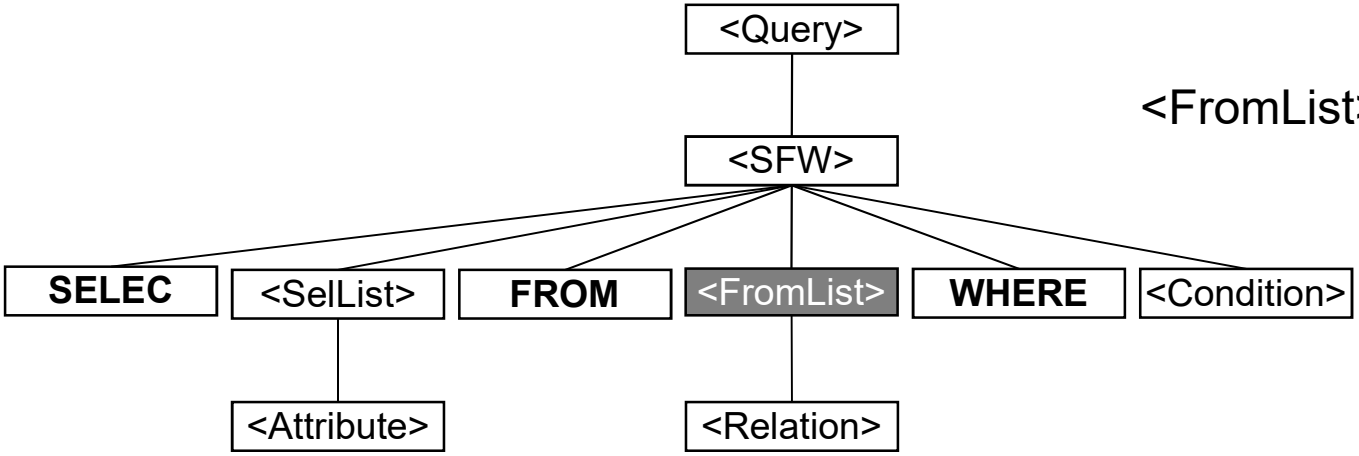

<SelList> ::= <Attribute>

# SQL Grammar

<Query> ::= <SFW>
<Query> ::= ( <Query> )
<SFW> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>
<SelList> ::= <Attribute>, <SelList>
<SelList> ::= <Attribute>
**<FromList> ::= <Relation>, <FromList>**
**<FromList> ::= <Relation>**
<Condition> ::= <Condition> AND <Condition>
<Condition> ::= <Condition> OR <Condition>
<Condition> ::= NOT <Condition>
<Condition> ::= <Tuple> IN <Query>
<Condition> ::= <Attribute> = <Attribute>
<Condition> ::= <Attribute> < <Attribute>
<Condition> ::= <Attribute> > <Attribute>
…
<Condition> ::= <Attribute> LIKE <Pattern>
<Condition> ::= <Attribute> EXIST <Pattern>
<Tuple> ::= <Attribute>

```
SELECT ID_S
FROM S
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500);
```

# Building the parse tree



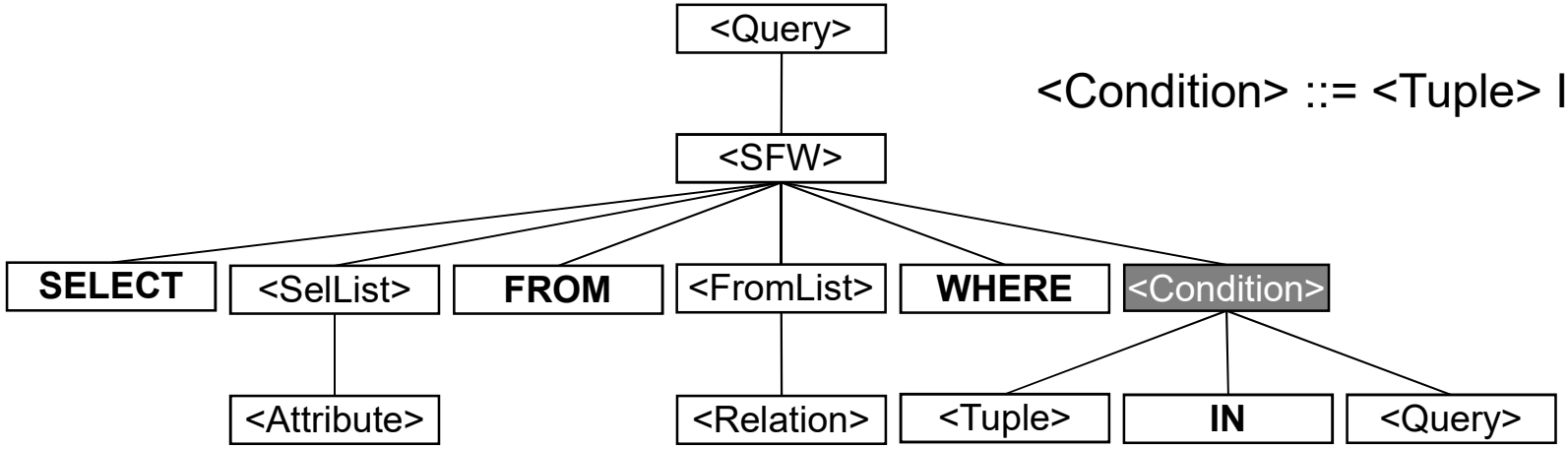

<FromList> ::= <Relation>

# SQL Grammar

<Query> ::= <SFW>
<Query> ::= ( <Query> )
<SFW> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>
<SelList> ::= <Attribute>, <SelList>
<SelList> ::= <Attribute>
<FromList> ::= <Relation>, <FromList>
<FromList> ::= <Relation>
**<Condition> ::= <Condition> AND <Condition>**
**<Condition> ::= <Condition> OR <Condition>**
**<Condition> ::= NOT <Condition>**
**<Condition> ::= <Tuple> IN <Query>**
**<Condition> ::= <Attribute> = <Attribute>**
**<Condition> ::= <Attribute> < <Attribute>**
**<Condition> ::= <Attribute> > <Attribute>**
**…**
**<Condition> ::= <Attribute> LIKE <Pattern>**
**<Condition> ::= <Attribute> EXIST <Pattern>**
<Tuple> ::= <Attribute>

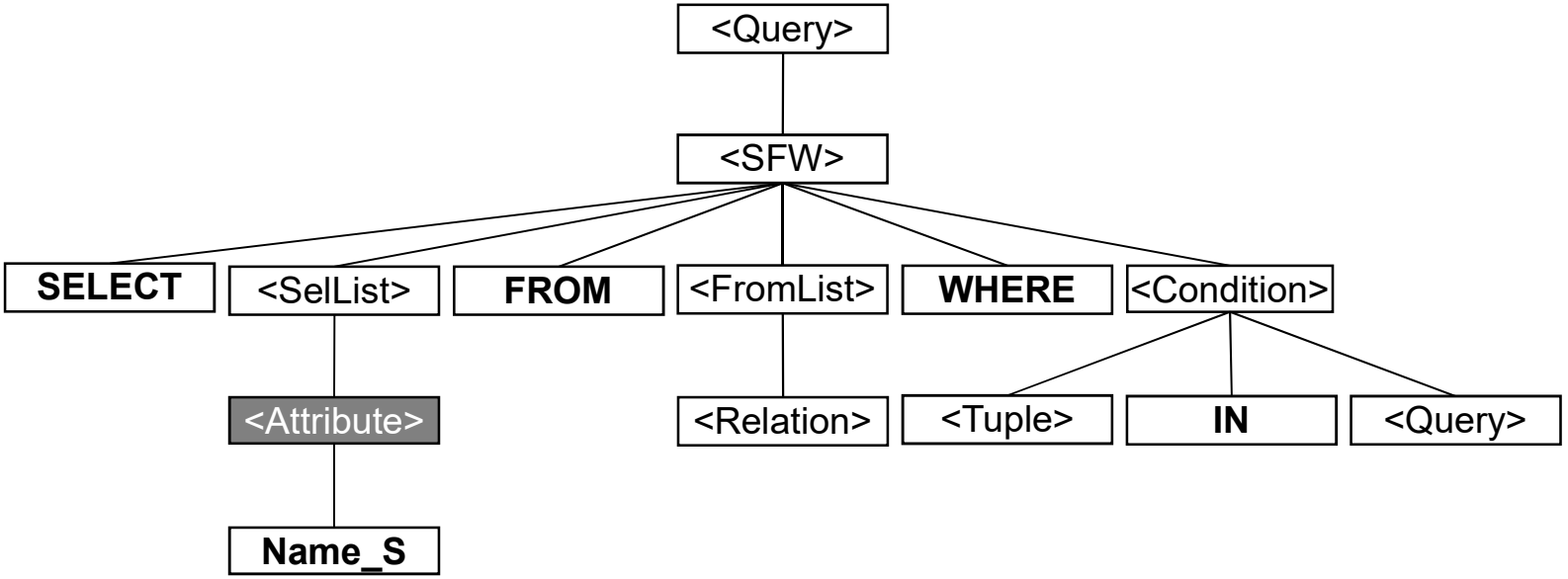
```
SELECT Name_S
FROM S
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500);
```

# Building the parse tree




<Condition> ::= <Tuple> IN <Query>

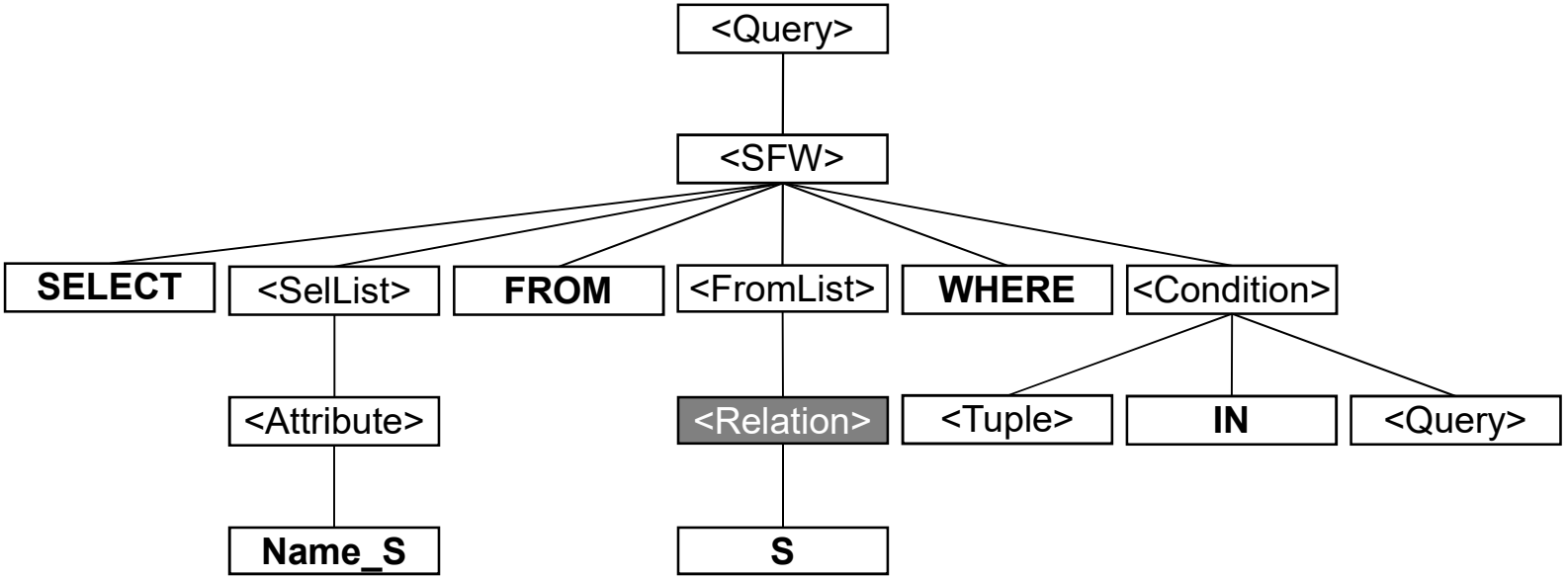
# Building the parse tree




<Attribute> = *identifier*

```
SELECT Name_S
FROM S
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500);
```

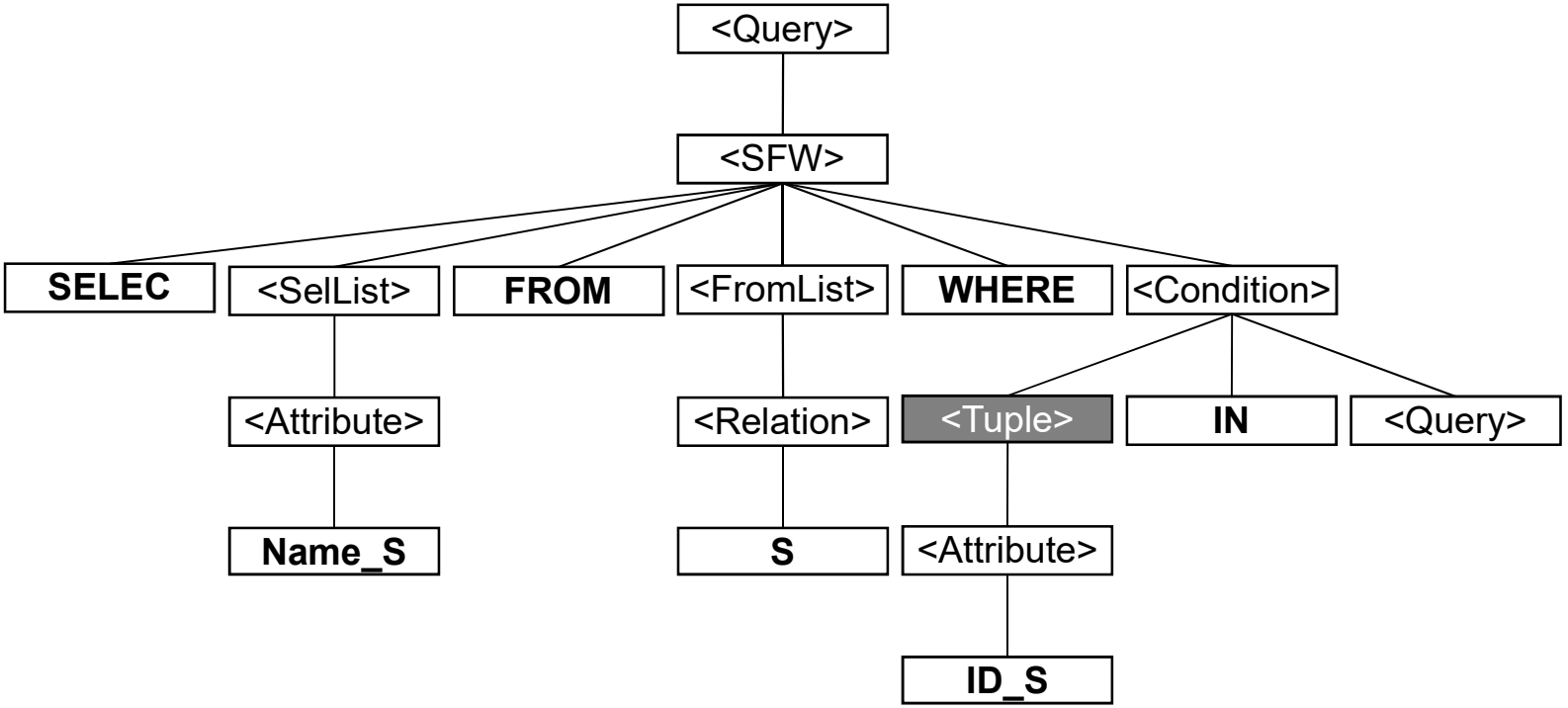
# Building the parse tree




<Relation> = *identifier*

```
SELECT ID_S
FROM S
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500);
```

# Building the parse tree

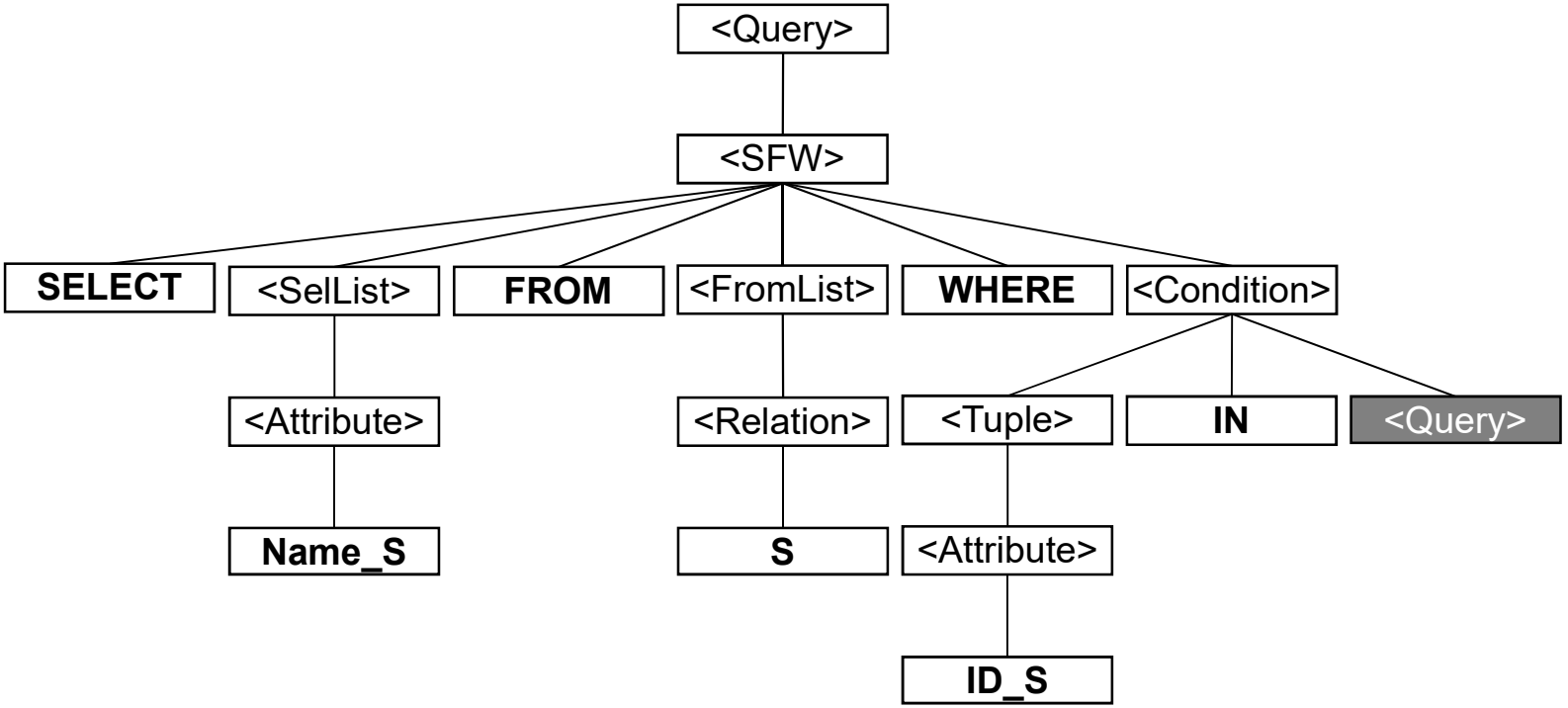



<Tuple> ::= <Attribute>

<Attribute> = *identifier*

```
SELECT ID_S
FROM
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500);
```

# Building the parse tree



# SQL Grammar

**<Query> ::= <SFW>**

**<Query> ::= ( <Query> )**

<SFW> ::= SELECT <SelList> FROM <FromList> WHERE <Condition>

<SelList> ::= <Attribute>, <SelList>

<SelList> ::= <Attribute>

<FromList> ::= <Relation>, <FromList>

<FromList> ::= <Relation>

<Condition> ::= <Condition> AND <Condition>

<Condition> ::= <Condition> OR <Condition>

<Condition> ::= NOT <Condition>

<Condition> ::= <Tuple> IN <Query>

<Condition> ::= <Attribute> = <Attribute>

<Condition> ::= <Attribute> < <Attribute>

<Condition> ::= <Attribute> > <Attribute>

…

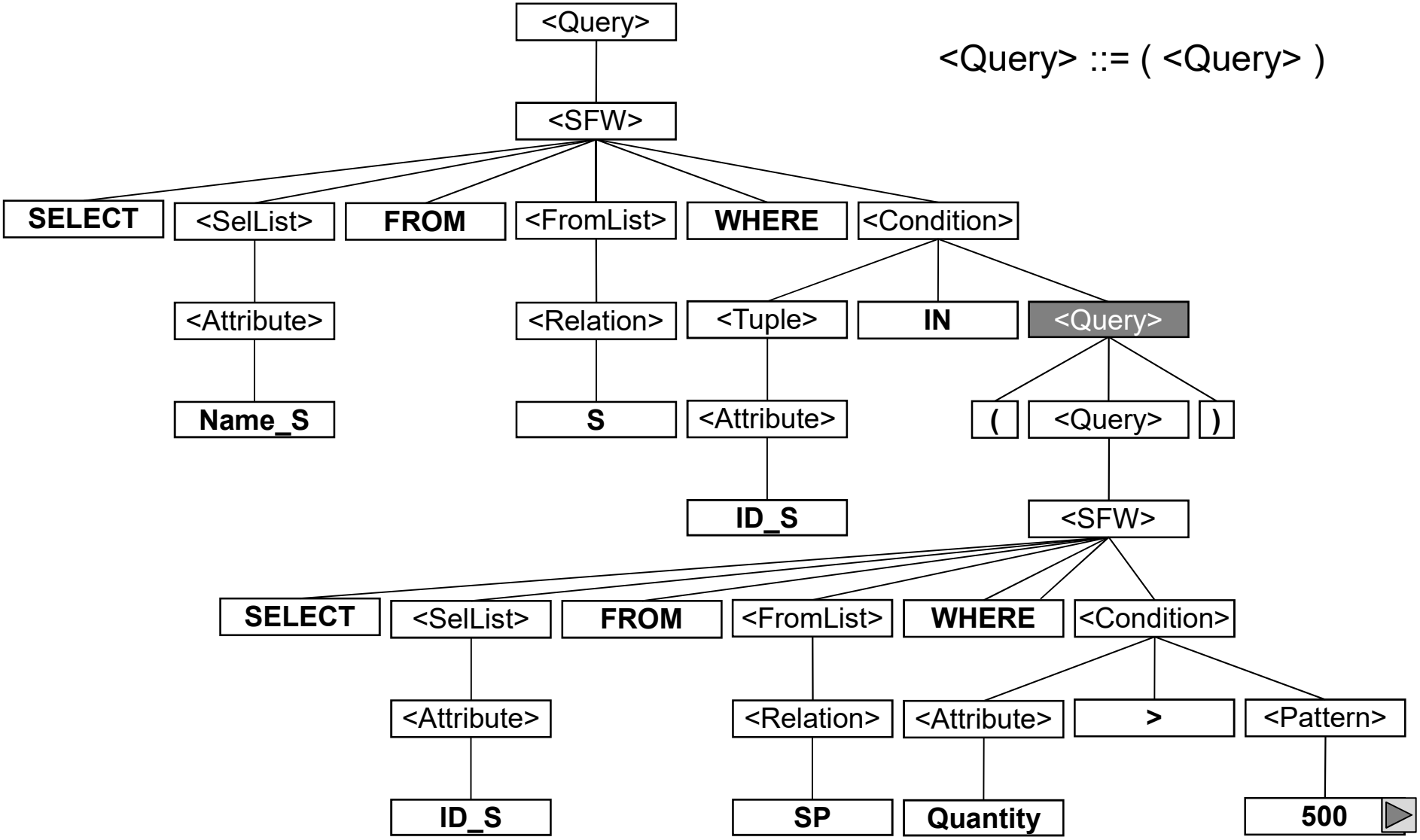
<Condition> ::= <Attribute> LIKE <Pattern>

<Condition> ::= <Attribute> EXIST <Pattern>

<Tuple> ::= <Attribute>

```
SELECT ID_S
FROM S
WHERE ID_S IN (
  SELECT ID_S
  FROM SP
  WHERE Quantity > 500)
```

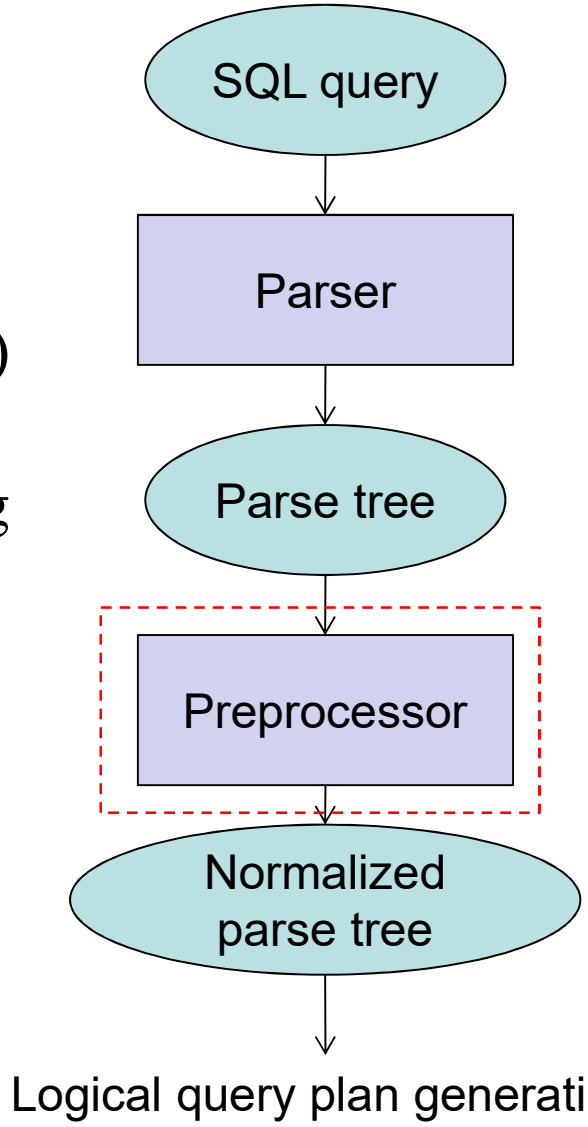
# Parse tree


<Query>
<Query> ::= ( <Query> )
<SFW>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<Relation>
<Tuple>
IN
<Query>
Name_S
S
<Attribute>
(
<Query>
)
ID_S
<SFW>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<Relation>
<Attribute>
>
<Pattern>
ID_S
SP
Quantity
500

# Query parsing outline

## Query parsing stages

1. Syntax analysis (performing by *Parser*)
2. View resolution and Semantic checking (performing by *Preprocessor*)



# Preprocessor

Preprocessor uses database schema.

# Database schema (metadata)

- Database schema is a collection of relation schemas
- Database schema is stored on the disk in the *Data dictionary*

# Schema of S (Suppliers) relation

Attributes of S (Suppliers)

| Identifier | Type | Length (bytes) | Restriction | Order number |
|---|---|---|---|---|
| City_S | Char | 16 | | 3 |
| Deposit | Float | 8 | | 5 |
| ID_S | Integer | 4 | Primary key | 1 |
| Name_S | Char | 32 | | 2 |
| Rating | Integer | 4 | | 4 |

# Preprocessor

- View resolution
- Semantic checking

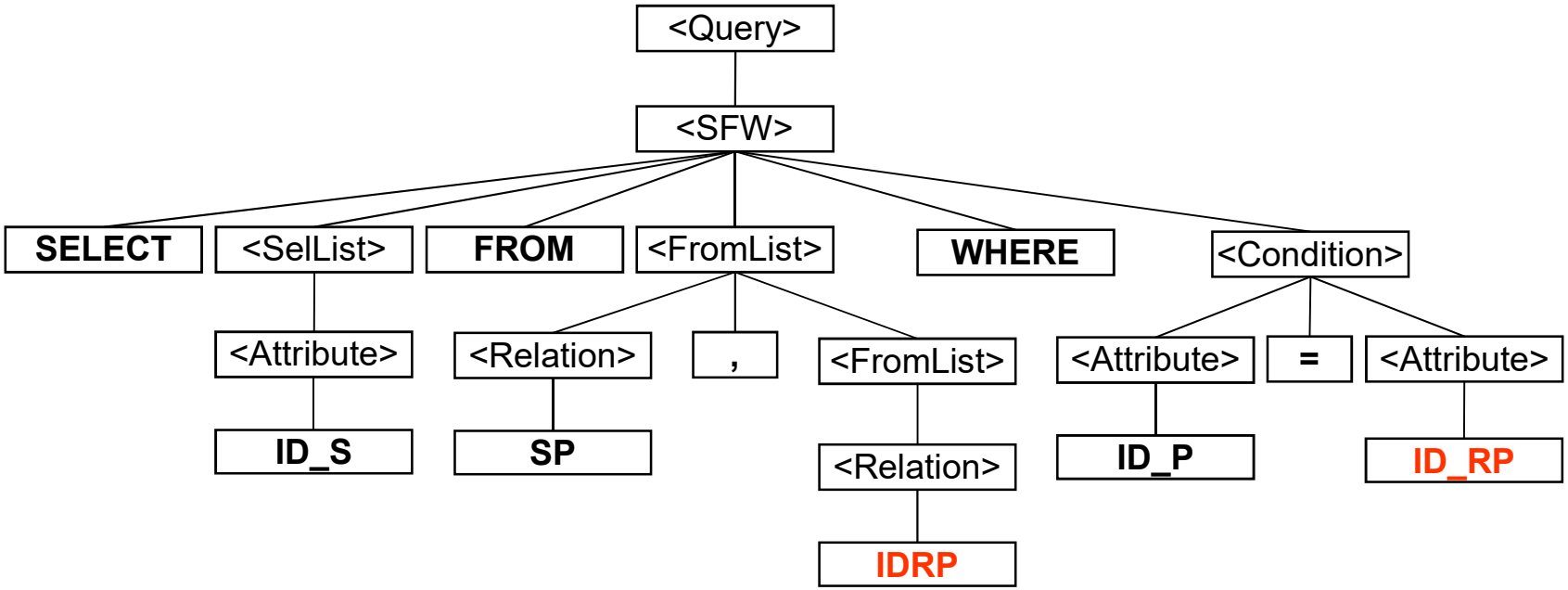
# View resolution

```
/* IDs of red parts */
CREATE VIEW IDRP (ID_RP)
   AS SELECT ID_P
   FROM P
   WHERE Color='Red';
…

/* IDs of suppliers which supply at least one red part */
SELECT ID_S
FROM SP, IDRP
WHERE ID_P= ID_RP;
```

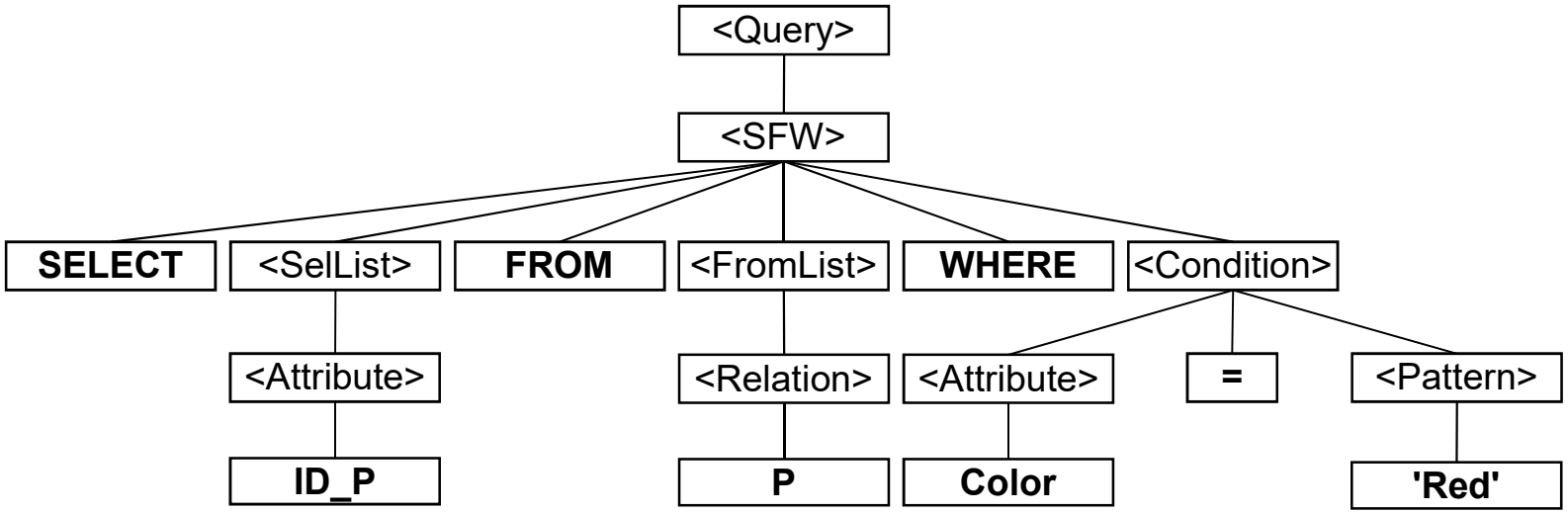
# Parse tree of the main query

```
/* IDs of suppliers which supply at least one red part */
SELECT ID_S
FROM SP, IDRP
WHERE ID_P= ID_RP;
```



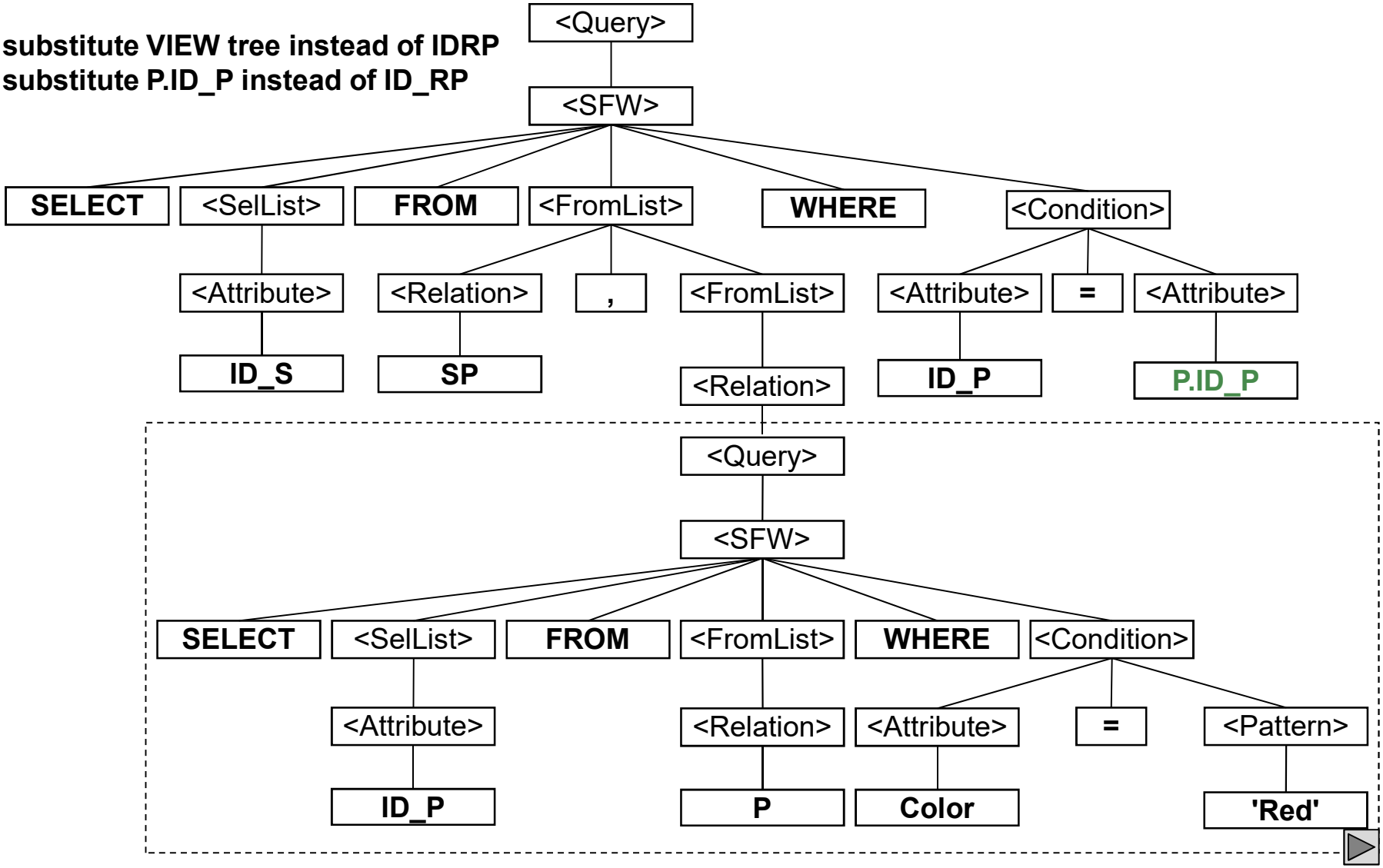
# Parse tree for the VIEW

```
/* IDs of red parts */
CREATE VIEW IDRP (ID_RP)
  AS SELECT ID_P
  FROM P
  WHERE Color='Red';
```



# Substitution

1. **To substitute VIEW tree instead of IDRP**
2. **To substitute P.ID_P instead of ID_RP**



# Semantic checking

1. ***Check relation uses.*** Every relation mentioned in a FROM-clause must be a relation in the current schema.
2. ***Check and resolve attribute uses***. Every attribute that is mentioned in the SELECT- or WHERE-clause must be an attribute of some relation in the current scope. The query processor would *resolve* each attribute by attaching to it the relation to which it refers, if that relation was not attached explicitly in the query. It would also check ambiguity, signaling an error if the attribute is in the scope of two or more relations with that attribute.
3. ***Check types***. All attributes must be of a type appropriate to their uses.

# Check and resolve attribute uses