FINAL PROJECT: SPECIAL-PURPOSE PROGRAMMING LANGUAGE (DSL)A

Salazar Andrés - 20202020043
Panqueva Miguel - 20201020174

INTRODUCTION

Relational databases are widely used to store and manage structured data in various applications. However, interacting with these databases often requires proficiency in SQL, which can be a barrier for users without technical expertise. To address this challenge, we developed a special-purpose programming language (DSL) that facilitates database design by transforming natural language instructions into SQL scripts.

EXAMPLE

```
ENTITY book :
    code: PK, NON_NULL, INT, AUT;
    author: NON_PK, NON_NULL, CHAR, NON_AUT;

ENTITY author :
    id: PK, NON_NULL, INT, NON_AUT;
    name: NON_PK, NON_NULL, CHAR, NON_AUT;

RELATIONSHIP write :
    author GO book : ONE_TO_MANY;
```

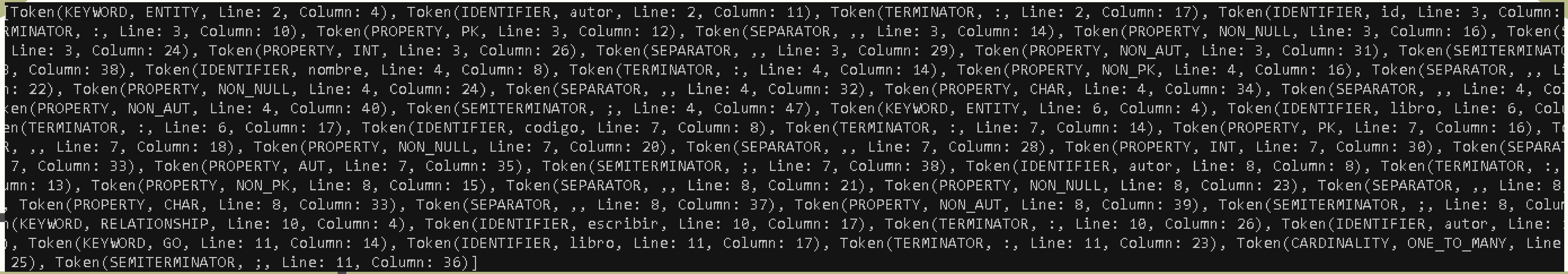
01

Token List

- KEYWORDS: ENTITY, RELATIONSHIP, GO
- CARDINALITY: ONE_TO_ONE, ONE_TO_MANY, MANY_TO_MANY
- PROPERTIES: PK, NON_PK, NON_NULL, NULL, INT, CHAR, AUT, NON_AUT
- IDENTIFIERS: Alphanumeric names for entities and attributes
- TERMINATORS: :
- SEPARATORS: ,
- SEMI-TERMINATORS: ;
- COMMENTS: //
- WHITESPACE: Spaces and tabs
- NEWLINES: \n
- MISMATCH: Any unrecognized character

LEXICAL ANALYZER

[Token(KEYWORD, ENTITY, Line: 2, Column: 4),
Token(IDENTIFIER, autor, Line: 2, Column: 11),
Token(TERMINATOR, :, Line: 2, Column: 17),
Token(IDENTIFIER, id, Line: 3, Column: 8),
Token(TERMINATOR, :, Line: 3, Column: 10),
Token(PROPERTY, PK, Line: 3, Column: 12),]



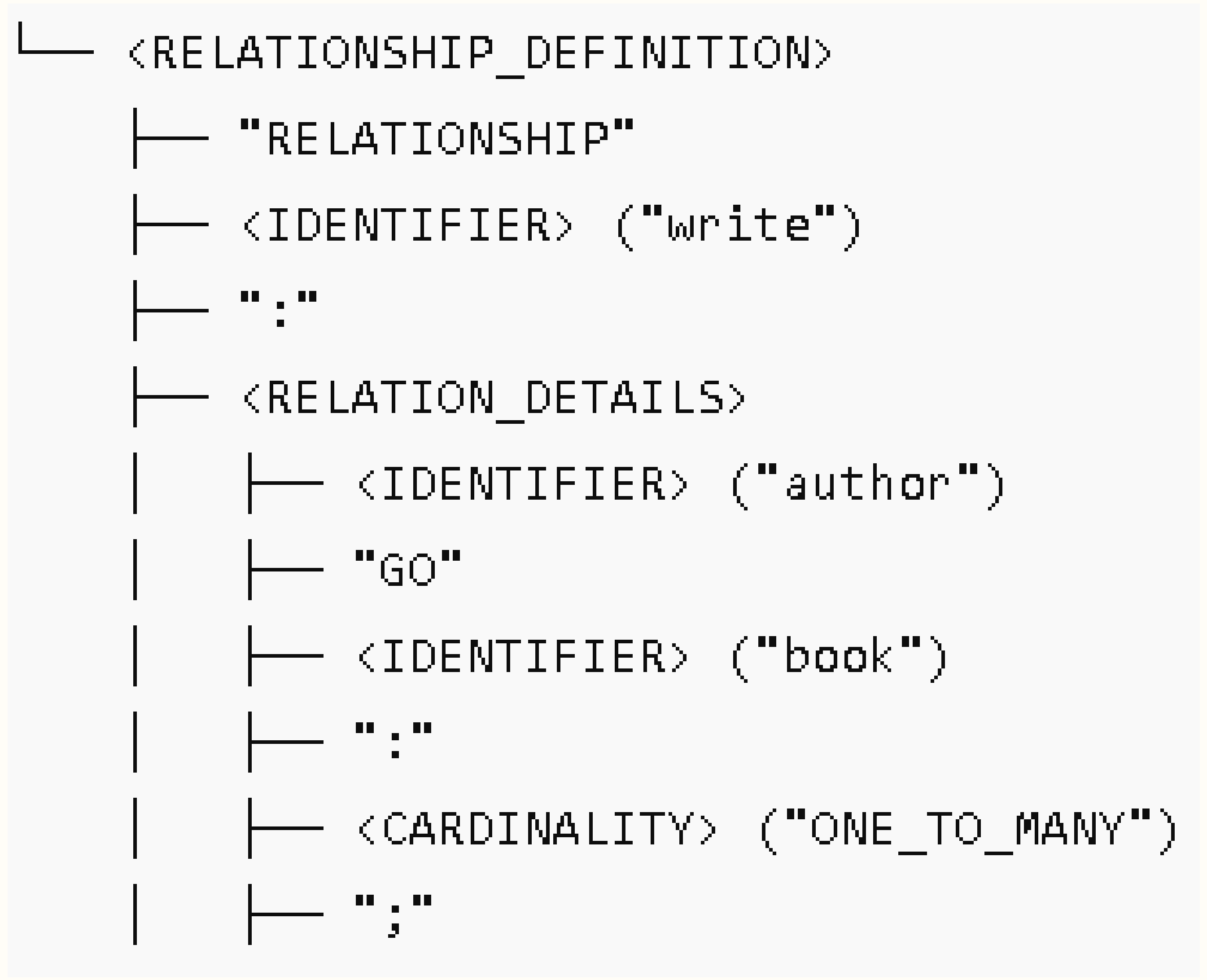
```
[Token(KEYWORD, ENTITY, Line: 2, Column: 4), Token(IDENTIFIER, autor, Line: 2, Column: 11), Token(TERMINATOR, :, Line: 2, Column: 17), Token(IDENTIFIER, id, Line: 3, Column:
RMINATOR, :, Line: 3, Column: 10), Token(PROPERTY, PK, Line: 3, Column: 12), Token(SEPARATOR, ,, Line: 3, Column: 14), Token(PROPERTY, NON_NULL, Line: 3, Column: 16), Token(S
 Line: 3, Column: 24), Token(PROPERTY, INT, Line: 3, Column: 26), Token(SEPARATOR, ,, Line: 3, Column: 29), Token(PROPERTY, NON_AUT, Line: 3, Column: 31), Token(SEMITERMINATO
3, Column: 38), Token(IDENTIFIER, nombre, Line: 4, Column: 8), Token(TERMINATOR, :, Line: 4, Column: 14), Token(PROPERTY, NON_PK, Line: 4, Column: 16), Token(SEPARATOR, ,, Li
n: 22), Token(PROPERTY, NON_NULL, Line: 4, Column: 24), Token(SEPARATOR, ,, Line: 4, Column: 32), Token(PROPERTY, CHAR, Line: 4, Column: 34), Token(SEPARATOR, ,, Line: 4, Co
ken(PROPERTY, NON_AUT, Line: 4, Column: 40), Token(SEMITERMINATOR, ;, Line: 4, Column: 47), Token(KEYWORD, ENTITY, Line: 6, Column: 4), Token(IDENTIFIER, libro, Line: 6, Colu
en(TERMINATOR, :, Line: 6, Column: 17), Token(IDENTIFIER, codigo, Line: 7, Column: 8), Token(TERMINATOR, :, Line: 7, Column: 14), Token(PROPERTY, PK, Line: 7, Column: 16), To
R, ,, Line: 7, Column: 18), Token(PROPERTY, NON_NULL, Line: 7, Column: 20), Token(SEPARATOR, ,, Line: 7, Column: 28), Token(PROPERTY, INT, Line: 7, Column: 30), Token(SEPARAT
 7, Column: 33), Token(PROPERTY, AUT, Line: 7, Column: 35), Token(SEMITERMINATOR, ;, Line: 7, Column: 38), Token(IDENTIFIER, autor, Line: 8, Column: 8), Token(TERMINATOR, :,
umn: 13), Token(PROPERTY, NON_PK, Line: 8, Column: 15), Token(SEPARATOR, ,, Line: 8, Column: 21), Token(PROPERTY, NON_NULL, Line: 8, Column: 23), Token(SEPARATOR, ,, Line: 8
, Token(PROPERTY, CHAR, Line: 8, Column: 33), Token(SEPARATOR, ,, Line: 8, Column: 37), Token(PROPERTY, NON_AUT, Line: 8, Column: 39), Token(SEMITERMINATOR, ;, Line: 8, Colu
n(KEYWORD, RELATIONSHIP, Line: 10, Column: 4), Token(IDENTIFIER, escribir, Line: 10, Column: 17), Token(TERMINATOR, :, Line: 10, Column: 26), Token(IDENTIFIER, autor, Line:
), Token(KEYWORD, GO, Line: 11, Column: 14), Token(IDENTIFIER, libro, Line: 11, Column: 17), Token(TERMINATOR, :, Line: 11, Column: 23), Token(CARDINALITY, ONE_TO_MANY, Line
 25), Token(SEMITERMINATOR, ;, Line: 11, Column: 36)]
```

Generative Diagram

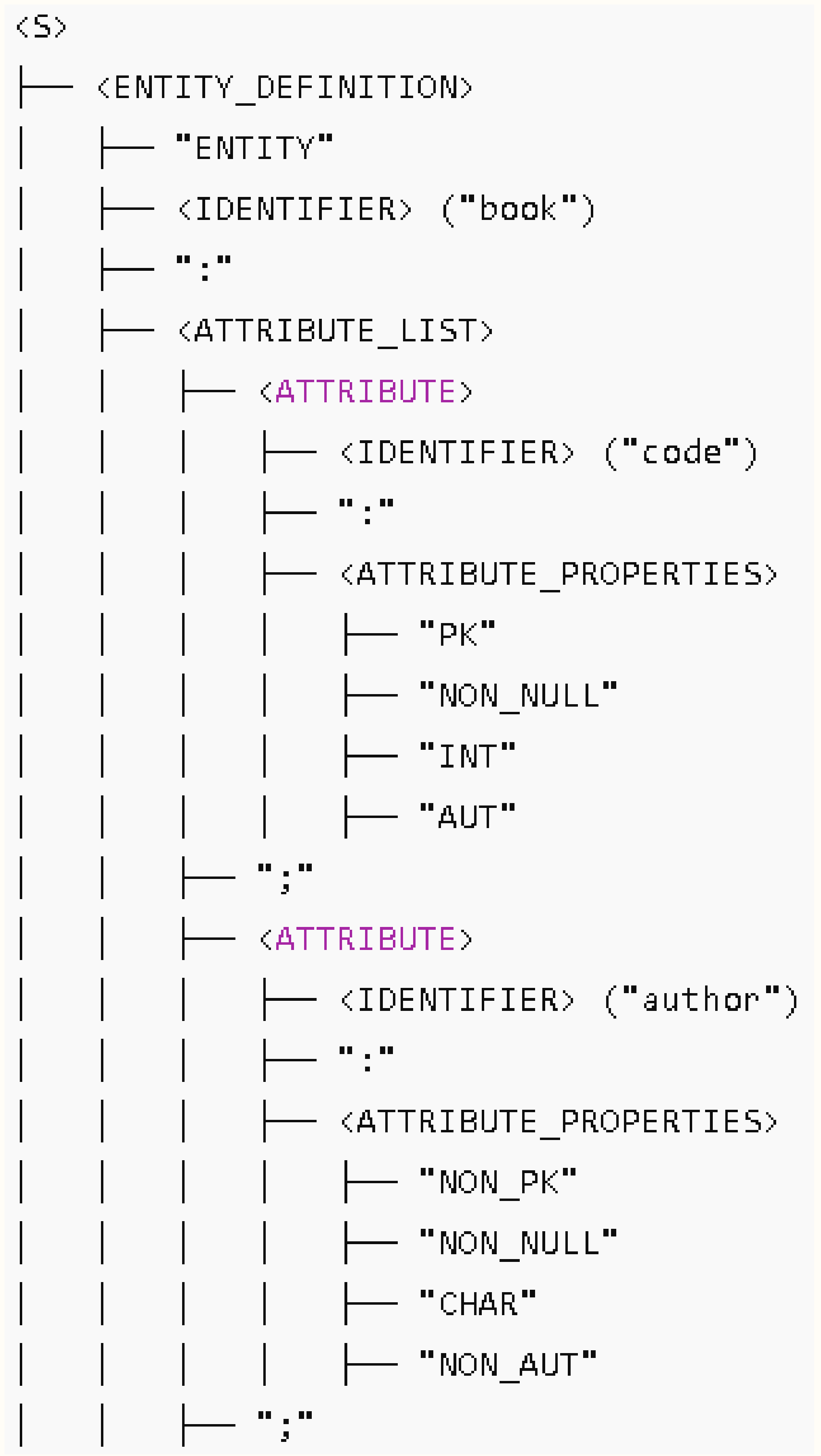
02

```
<S>                           -> <ENTITY_DEFINITION> | <RELATIONSHIP_DEFINITION>
<ENTITY_DEFINITION>           -> "ENTITY" <IDENTIFIER> ":" <ATTRIBUTE_LIST>
<ATTRIBUTE_LIST>              -> <ATTRIBUTE> ";" | <ATTRIBUTE> ";" <ATTRIBUTE_LIST>
<ATTRIBUTE>                   -> <IDENTIFIER> ":" <ATTRIBUTE_PROPERTIES>
<ATTRIBUTE_PROPERTIES>        -> <PROPERTY> | <PROPERTY> "," <ATTRIBUTE_PROPERTIES>
<PROPERTY>                    -> "PK" | "NON_PK" | "NON_NULL" | "NULL" | "INT" | "CHAR" | "AUT" | "NON
<RELATIONSHIP_DEFINITION> -> "RELATIONSHIP" <IDENTIFIER> ":" <RELATION_DETAILS>
<RELATION_DETAILS>            -> <IDENTIFIER> "GO" <IDENTIFIER> ":" <CARDINALITY> ";"
<CARDINALITY>                 -> "ONE_TO_ONE" | "ONE_TO_MANY" | "MANY_TO_MANY"
```

Applying generative diagram to the example



```
└── <RELATIONSHIP_DEFINITION>
    ├── "RELATIONSHIP"
    ├── <IDENTIFIER> ("write")
    ├── ":"
    ├── <RELATION_DETAILS>
    │   ├── <IDENTIFIER> ("author")
    │   ├── "GO"
    │   ├── <IDENTIFIER> ("book")
    │   ├── ":"
    │   ├── <CARDINALITY> ("ONE_TO_MANY")
    │   ├── ";"
```



```
<S>
├── <ENTITY_DEFINITION>
│   ├── "ENTITY"
│   ├── <IDENTIFIER> ("book")
│   ├── ":"
│   ├── <ATTRIBUTE_LIST>
│   │   ├── <ATTRIBUTE>
│   │   │   ├── <IDENTIFIER> ("code")
│   │   │   ├── ":"
│   │   │   ├── <ATTRIBUTE_PROPERTIES>
│   │   │   │   ├── "PK"
│   │   │   │   ├── "NON_NULL"
│   │   │   │   ├── "INT"
│   │   │   │   ├── "AUT"
│   │   ├── ";"
│   │   ├── <ATTRIBUTE>
│   │   │   ├── <IDENTIFIER> ("author")
│   │   │   ├── ":"
│   │   │   ├── <ATTRIBUTE_PROPERTIES>
│   │   │   │   ├── "NON_PK"
│   │   │   │   ├── "NON_NULL"
│   │   │   │   ├── "CHAR"
│   │   │   │   ├── "NON_AUT"
│   │   ├── ";"
```

semantic analyzer

DUPLICITY

When there are two entities, attributes or relationships with the same identifier

LOGICAL INCONSISTENCY

when a set of rules, statements, or data in a system conflict, that is, when two or more statements contradict each other

INCOMPLETE RELATIONSHIP

When one of the entities for the relationship is missing

INCOMPLETE INFORMATION

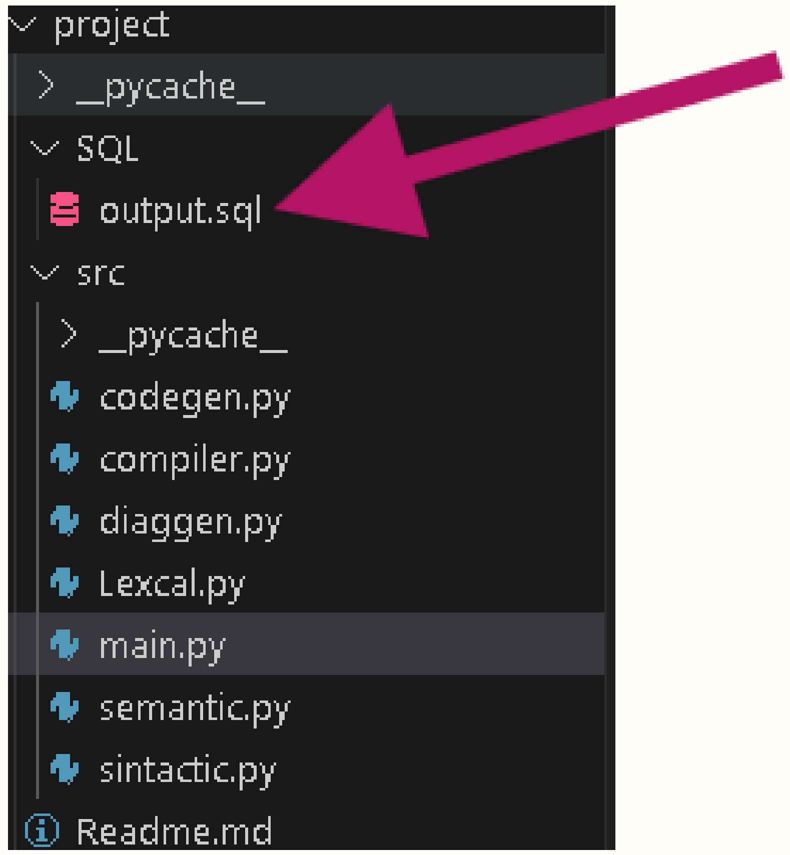
It is requested that there be 4 properties for each attribute. In case one or more is missing, this is presented

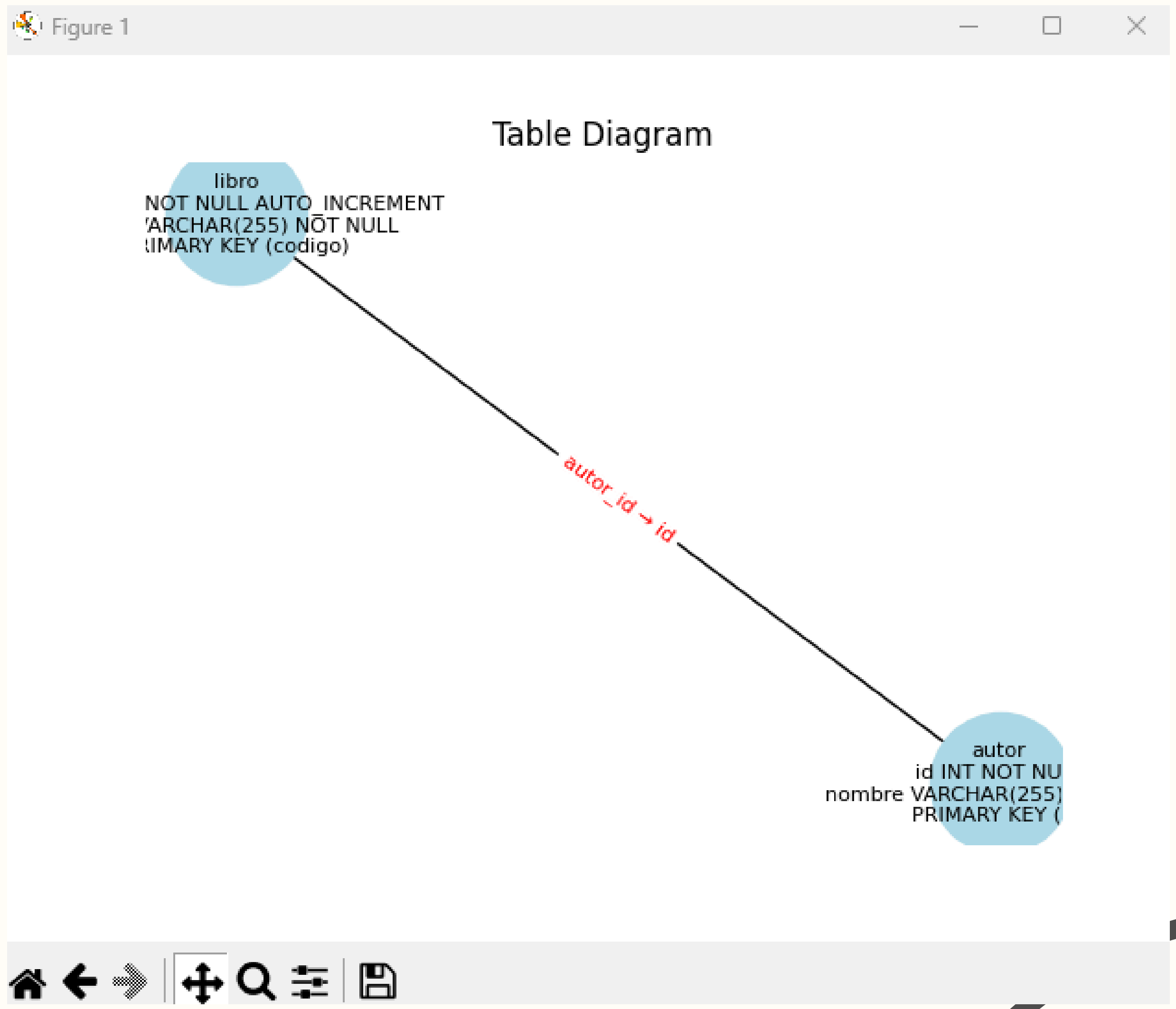
CARDINALITY

Check that there is a cardinality in the database

Final results

1. SQL File: Generates an output.sql file with the database structure.
2. E-R Diagram: Generates a simple graphical representation using matplotlib and networkx.





THANK YOU SO MUCH