Universidad Autónoma de Zacatecas

Academic Unit of Electrical Engineering

Software Engineering Academic Program

Group: 5B - Semester: 2022-5º

Practice Number: 05

Practice Name: DDL

Date: 15/September/2022

Professor:

Aldonso Becerra Sánchez.

Student:

Cristian Omar Alvarado Rodríguez.

Índice

DDL2

September 8th, 2022

1. Introduction

Data Definition Language (DDL) is a subset of SQL. It is a language for describing data and their relationships in a database. You can generate DDL in a database object script to:

- Keep a snapshot of the database structure.

- Set up a test system where the database acts like the production system, but contains no data.

- Produce templates for new objects that you can create based on existing ones.

DDL statements are used to describe a database, to define its structure, to create its objects, and to create the subobjects of the table. You can make changes to a rule set after you create it.

Today's database industry embeds DDL in any formal language that describes data. However, it is considered a subset of SQL (Structured Query Language). SQL often uses normal English imperative verbs as sentences to implement modifications to the database. Therefore, DDL does not appear as a different language in an SQL database, but it does define changes to the database schema.

2. Practice objective

Review the notion of Oracle DDL statements creating corrections of co-workers.

3. Developing

Activity 1: You should define a problem statement about a topic of interest (a brief description). Write it as part of the activity 1.

"An auto insurer wants to keep track of its customers as well as the cars insured for each customer. For the extraction of data, consider the insurance policy of the clients, said policy must contain both information about the client, the car and the sales agent who made it."

Activity 2: The problem statement of activity 1 will be passed to you (from another classmate). With this problem statement, you should be able to generate the ER diagram.

The following statement was passed on to me by my classmate Alan Martín Romo Aréchiga.

"A tennis store wants to save in a database the tennis models they handle, for this they want to save the model, the brand, the color, material, price and the size number of said pair. You also want to have a record of the employees who work in the store and the sales that are made."

Based on the previous statement, the entity-relationship diagram shown in **Figure 1** was made.

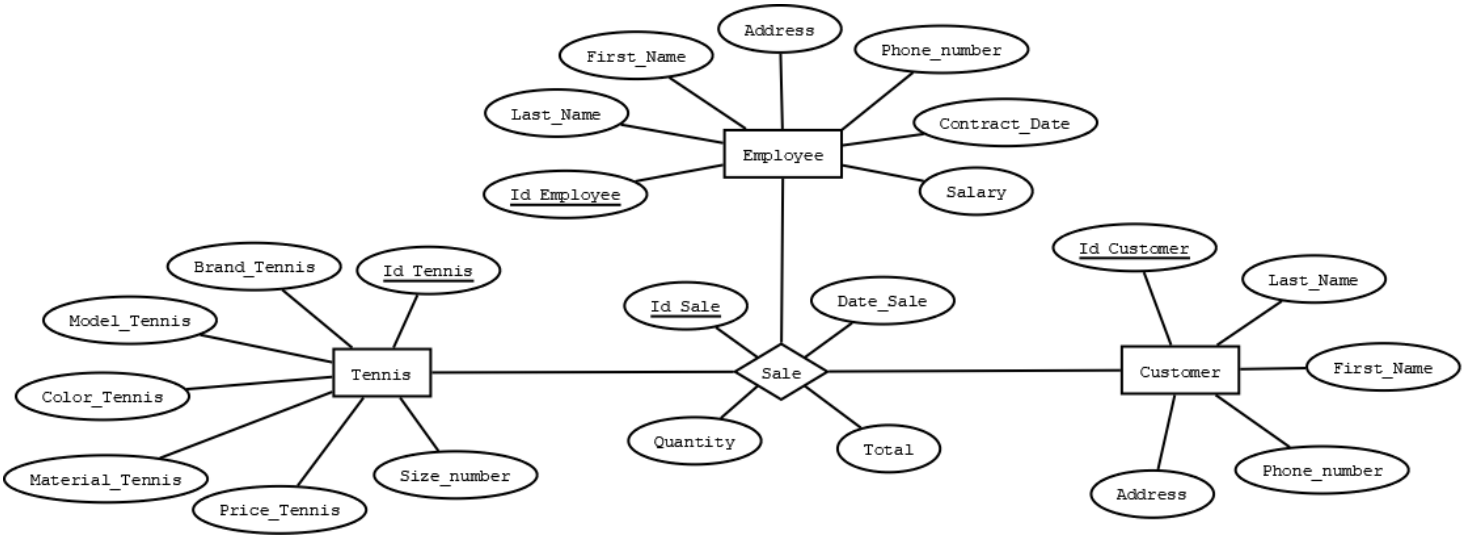


Figura 1: *Entity-relationship diagram*

Activity 3: The problem statement and its corresponding ER diagram of activity 2 will be passed to you (from another classmate). With these two items, you should correct the necessary parts of the ER diagram (using your abstraction) according to its problem statement, then, you should be able to generate the relational diagram by using "Dia" software.

The following entity-relationship diagram of **figure 2** was passed to me by my classmate Alan Martín Romo Aréchiga as well as his statement shown below.

"A phone factory needs a tool capable of managing the company's data, such as the monthly supplies received at the different warehouses, the employees who work, the products handled by the factory and the department stores that receive these products."

I made a few small corrections to the statement since I noticed that it had ambiguities and needed to clarify the ideas a little more.

"A phone factory needs to manage its data, such as the monthly inputs that are received in the different warehouses, the employees that work, the phones that the factory handles, and the department stores that receive these phones."

Some things were corrected to the entity-relationship diagram, such as: the FABRIC_RECEIVES relationship was removed, since this relationship could be grouped in an aggregation to make it more understandable, since the central idea was that the supplies would be stored in the warehouses so that later they could be They can manufacture the telephones so that they can later be sent to the different stores. New attributes were also added and some relations were renamed.

Figure 3 shows the new entity-relationship diagram that was made.

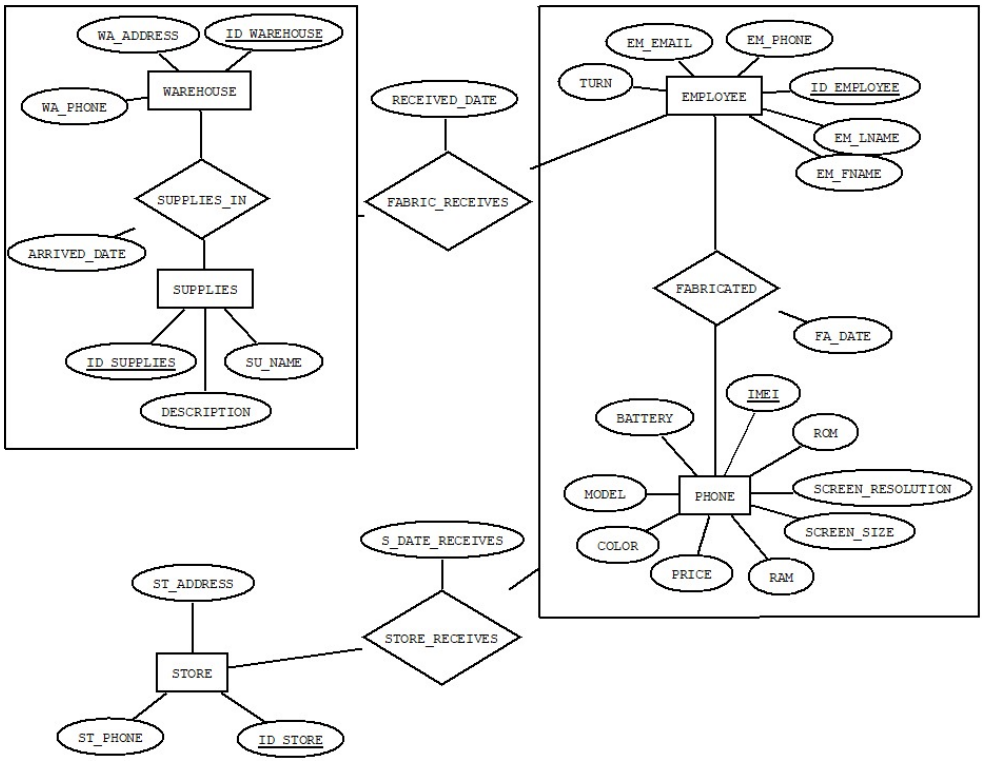


Figura 2: *Entity-relationship diagram*

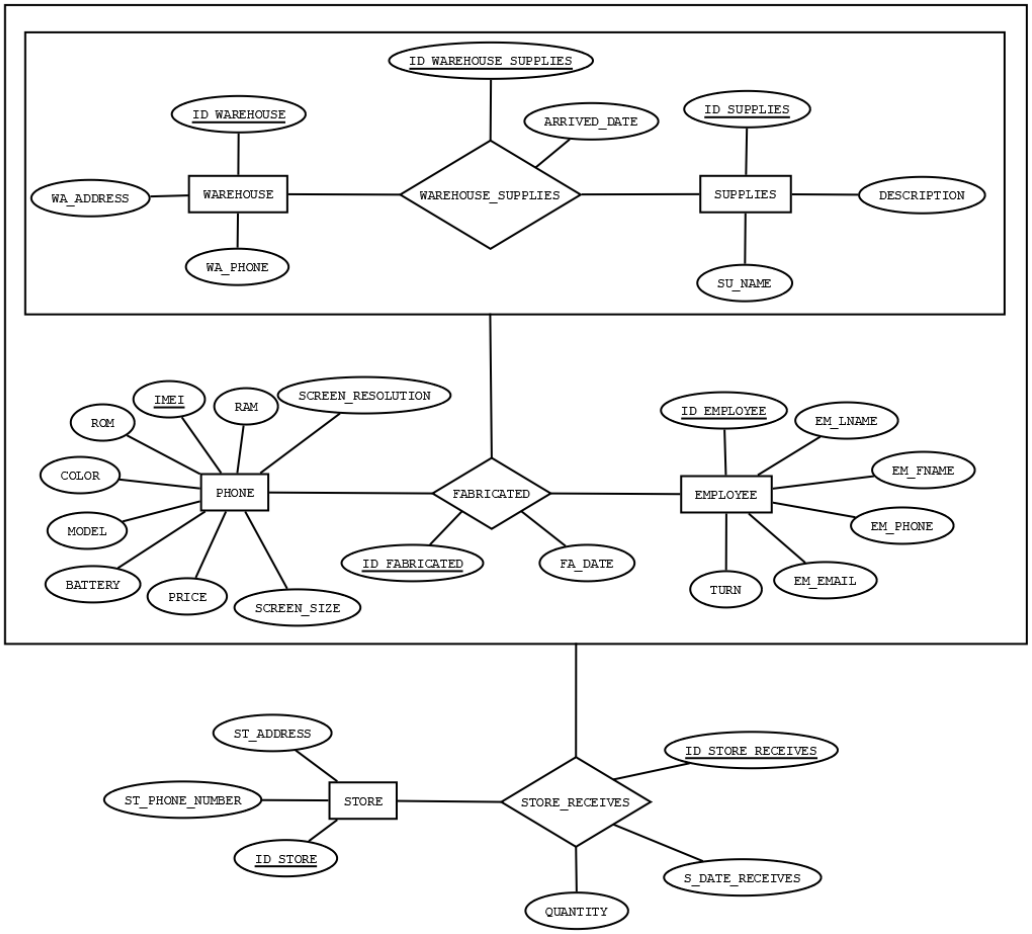


Figura 3: *Entity-relationship diagram*

Once the entity-relationship diagram was corrected, the relational model was made, which can be seen in **figure 4.**

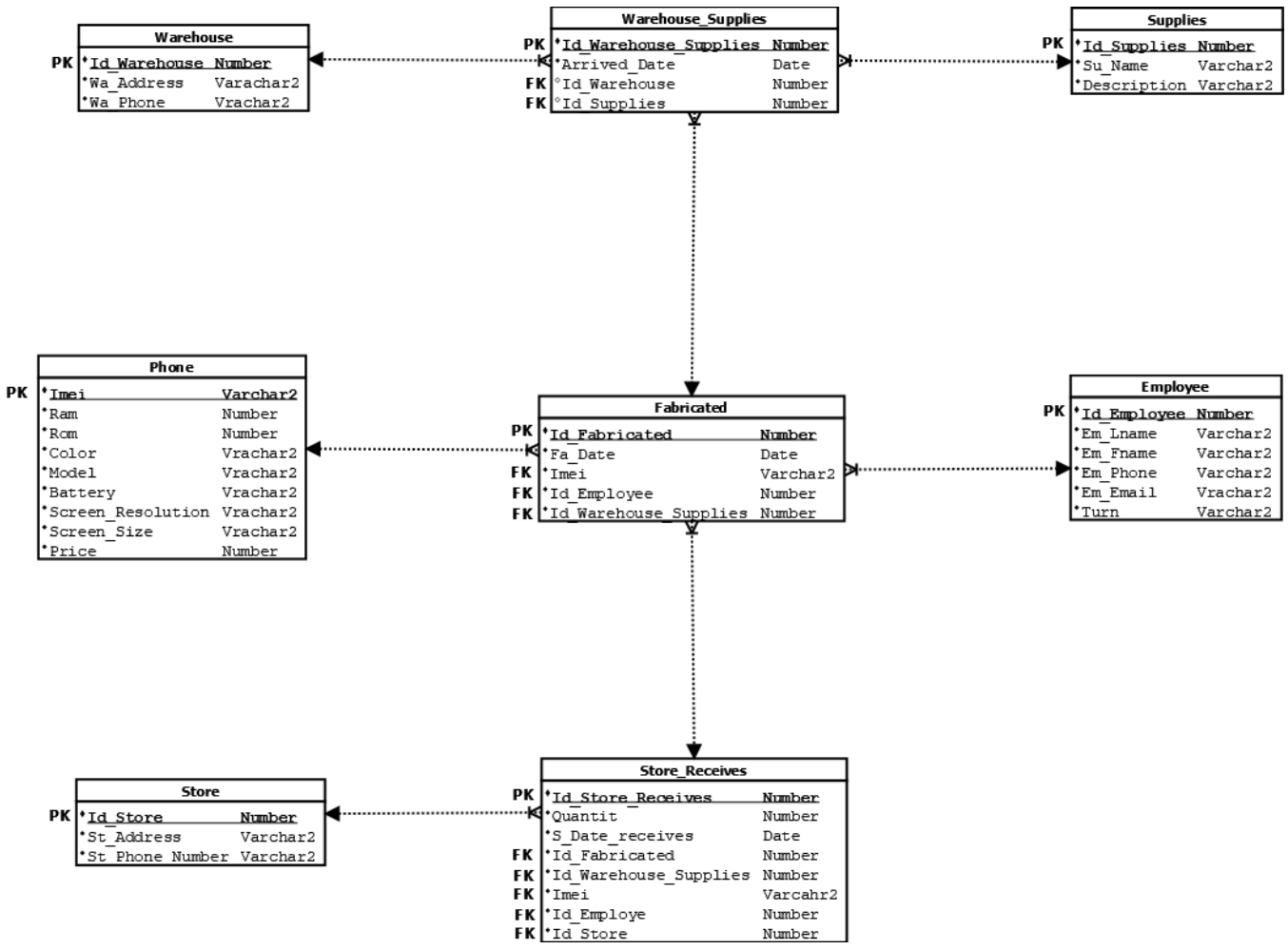


Figura 4: *Relational model of activity three*

Activity 4: The ER and relational diagrams of activity 3 will be passed to you (from another classmate). With just these two diagrams, you should correct the necessary parts of the relational diagram (using your abstraction) according to its ER diagram, then you should be able to generate the Oracle DDL sentences. You should add the basic indexes according to the diagram (reading the possible data to extract), the necessary sequences for the solution and the appropriate synonyms considering your insight.

Note: use the recommended execution order for the corresponding DDL sentences.

With these tables, you should automatically generate the physical diagram in DATA MODELER (dragging the tables). Compare this diagram with the relational model made by Dia.

The following entity-relationship diagram shown in **figure 5** was passed to me by my classmate Alan Martín Romo Aréchiga.

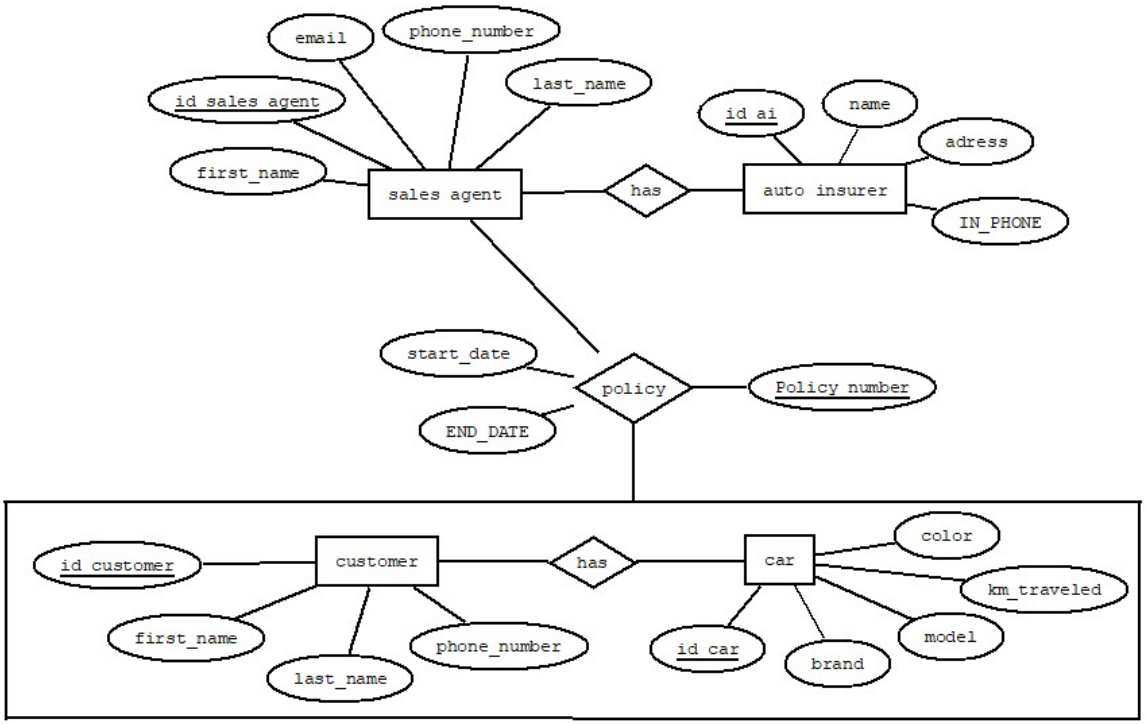


Figura 5: *Entity-relationship diagram*

The relational model shown in **figure 6** was passed on to me by my classmate Alan Martín Romo Aréchiga.

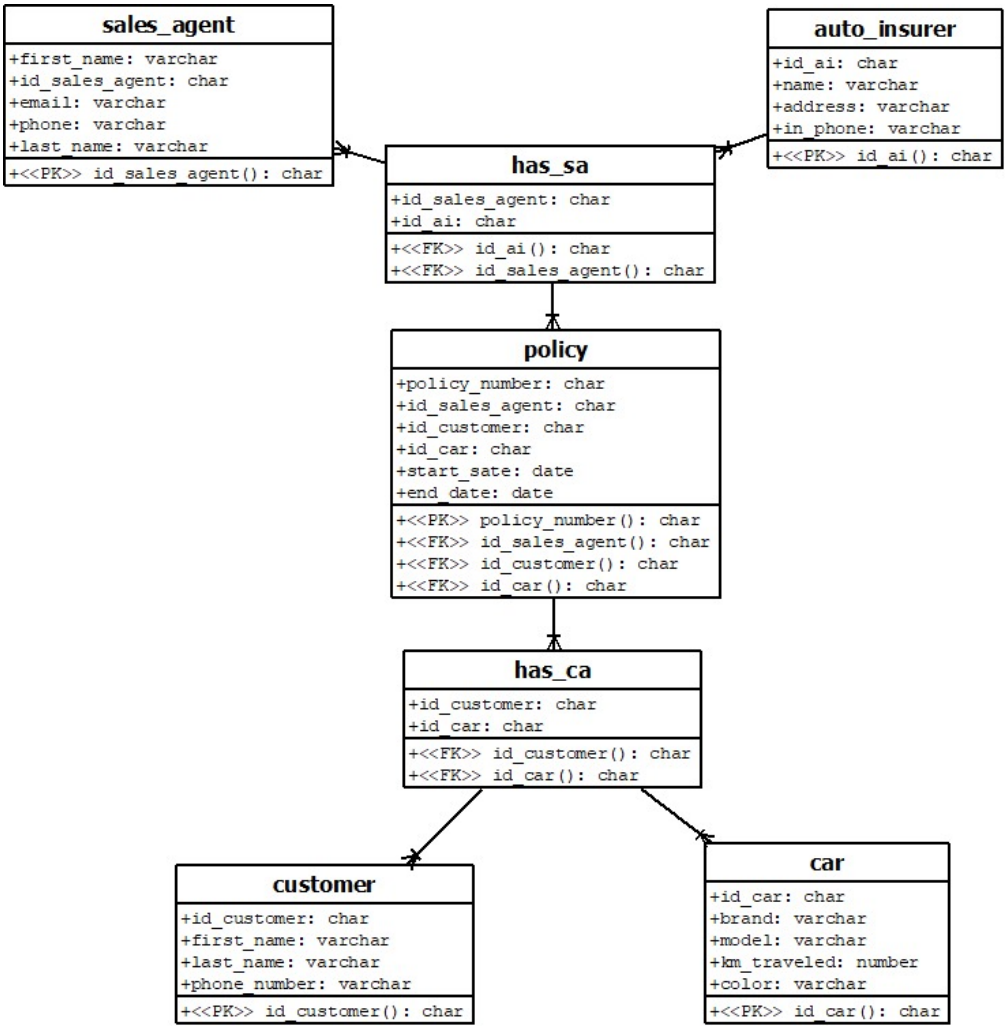


Figura 6: *Relational model*

The following **figure 7** shows the corrected relational model, the changes that were made were that the primary keys changed the data type from char to number, also the has_sa and has_ca tables had their respective primary keys added. since they did not have one, another change that was made was the change of the arrows since the correct symbolism was not being used correctly.

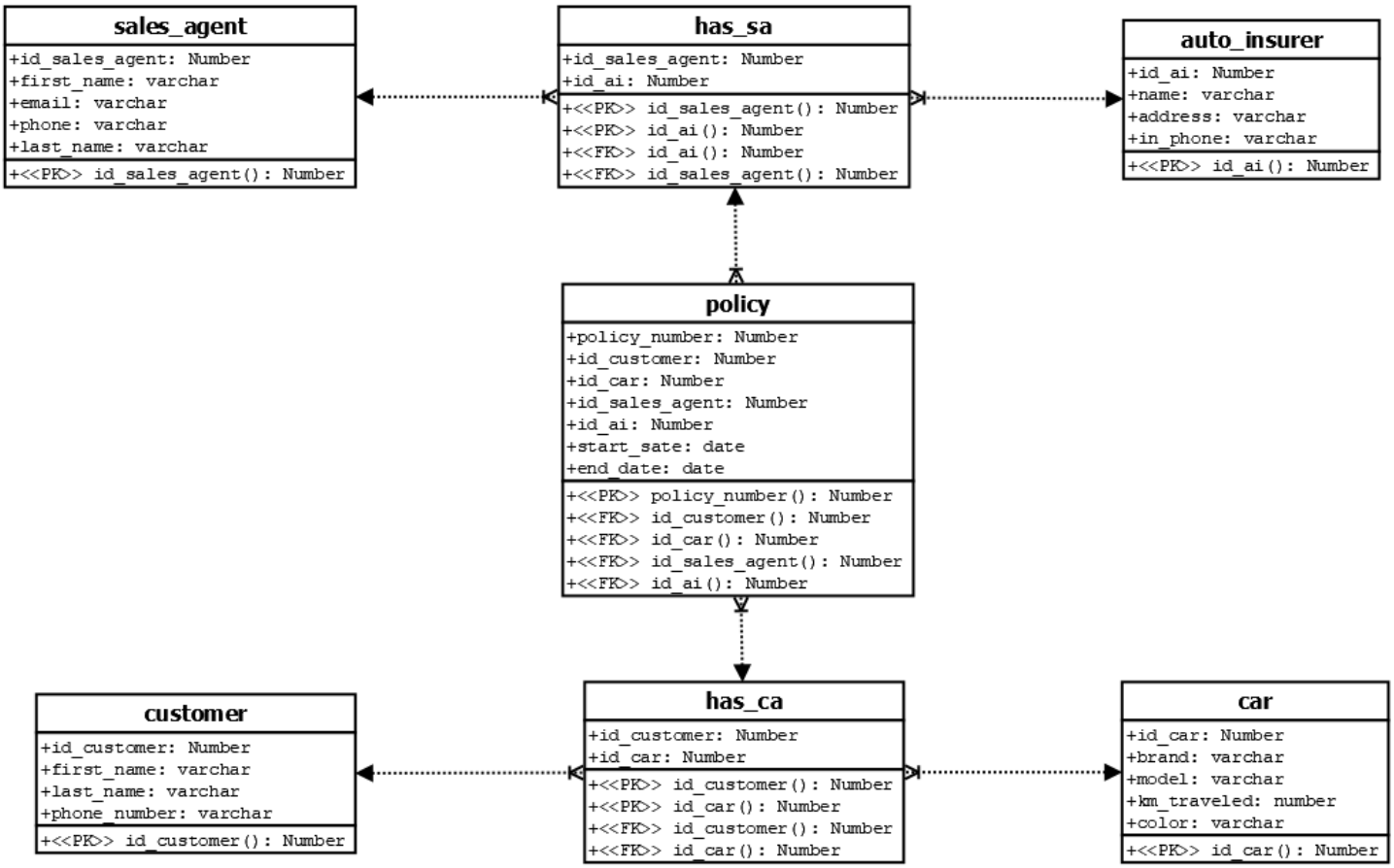
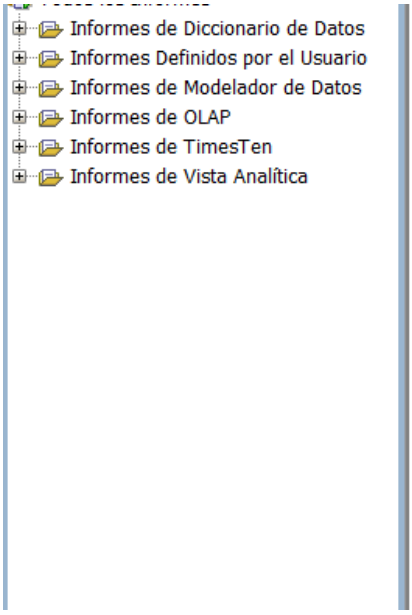


Figura 7: *Relational model*

In **figures 8 and 9** you can see the DDL statements for the creation of the different tables of the relational model.

Figura 8: *DDL statements, creation of tables*




```
CREATE TABLE HAS_SA (
    ID_SALES_AGENET NUMBER(5),
    ID_AI NUMBER(5)
);

CREATE TABLE POLICY (
    POLICY_NUMBER NUMBER(6),
    ID_CUSTOMER NUMBER(5),
    ID_CAR NUMBER(5),
    ID_SALES_AGENET NUMBER(5),
    ID_AI NUMBER(5),
    START_SATE DATE,
    END_DATE DATE
);
```

Figura 9: *DDL statements, creation of tables*

The following **figure 10** shows the DDL statements for the creation of the indexes for the columns of the tables.

```
-- INDEX CREATION
CREATE INDEX CUSTOMER_ID_IDX ON CUSTOMER (ID_CUSTOMER);
CREATE INDEX CUSTOMER_NAME_IDX ON CUSTOMER (FIRST_NAME, LAST_NAME);
CREATE INDEX CAR_ID_IDX ON CAR (ID_CAR);
CREATE INDEX HAS_CA_CUS_CAR_IDX ON HAS_CA (ID_CUSTOMER, ID_CAR);
CREATE INDEX SALES_AGENET_ID_IDX ON SALES_AGENET (ID_SALES_AGENET);
CREATE INDEX SALES_AGENET_NAME_IDX ON SALES_AGENET (FIRST_NAME, LAST_NAME);
CREATE INDEX AUTO_INSURER_ID_IDX ON AUTO_INSURER (ID_AI);
CREATE INDEX AUTO_INSURER_NAME_IDX ON AUTO_INSURER (NAME);
CREATE INDEX HAS_SA_AGEN_AI_IDX ON HAS_SA (ID_SALES_AGENET, ID_AI);
CREATE INDEX POLICY_NUMBER_IDX ON POLICY (POLICY_NUMBER);
```

Figura 10: *DDL statements, creation of indexes.*

Figure 11 shows the DDL statements for creating the primary keys and foreign keys of the tables.

```
-- PRIMARY KEYS
ALTER TABLE CUSTOMER ADD CONSTRAINT CUSTOMER_PK PRIMARY KEY (ID_CUSTOMER);
ALTER TABLE CAR ADD CONSTRAINT CAR_PK PRIMARY KEY (ID_CAR);
ALTER TABLE HAS_CA ADD CONSTRAINT HAS_CA_PK PRIMARY KEY (ID_CUSTOMER, ID_CAR);
ALTER TABLE SALES_AGENET ADD CONSTRAINT SALES_AGENET_PK PRIMARY KEY (ID_SALES_AGENET);
ALTER TABLE AUTO_INSURER ADD CONSTRAINT AUTO_INSURER_PK PRIMARY KEY (ID_AI);
ALTER TABLE HAS_SA ADD CONSTRAINT HAS_SA_PK PRIMARY KEY (ID_SALES_AGENET, ID_AI);
ALTER TABLE POLICY ADD CONSTRAINT POLICY_PK PRIMARY KEY (POLICY_NUMBER);

-- FOREIGN KEYS
ALTER TABLE HAS_CA ADD CONSTRAINT CUSTOMER_HAS_CA_FK FOREIGN KEY (ID_CUSTOMER)
            REFERENCES CUSTOMER (ID_CUSTOMER);

ALTER TABLE HAS_CA ADD CONSTRAINT CAR_HAS_CA_FK FOREIGN KEY (ID_CAR)
            REFERENCES CAR (ID_CAR);

ALTER TABLE HAS_SA ADD CONSTRAINT SALES_AGEN_HAS_SA_FK FOREIGN KEY (ID_SALES_AGENET)
            REFERENCES SALES_AGENET (ID_SALES_AGENET);

ALTER TABLE HAS_SA ADD CONSTRAINT AI_HAS_SA_FK FOREIGN KEY (ID_AI)
            REFERENCES AUTO_INSURER (ID_AI);

ALTER TABLE POLICY ADD CONSTRAINT HAS_CA_POLICY_FK FOREIGN KEY (ID_CUSTOMER, ID_CAR)
            REFERENCES HAS_CA (ID_CUSTOMER, ID_CAR);

ALTER TABLE POLICY ADD CONSTRAINT HAS_SA_POLICY_FK FOREIGN KEY (ID_SALES_AGENET, ID_AI)
            REFERENCES HAS_SA (ID_SALES_AGENET, ID_AI);
```

Figura 11: *DDL statements, creation of primary keys and foreign keys.*

Figure 12 shows the DDL statements for creating the sequences for the primary keys of the tables.

```
-- CREATE SEQUENCES
CREATE SEQUENCE CUSTOMER_ID_SEQ
                INCREMENT BY 1
                START WITH 1;

CREATE SEQUENCE CAR_ID_SEQ
                INCREMENT BY 1
                START WITH 1;

CREATE SEQUENCE SALES_AGENET_ID_SEQ
                INCREMENT BY 1
                START WITH 1;

CREATE SEQUENCE AUTO_INSURER_ID_SEQ
                INCREMENT BY 1
                START WITH 1;

CREATE SEQUENCE POLICY_ID_SEQ
                INCREMENT BY 1
                START WITH 1;
```

Figura 12: *DDL Statements, Sequence Creation.*

The following **figure 13** shows the DDL statements for the creation of synonyms for the tables.

```
-- CREATE SYNONYMS
CREATE SYNONYM CUS FOR CUSTOMER;
CREATE SYNONYM H_CA FOR HAS_CA;
CREATE SYNONYM SA FOR SALES_AGENET;
CREATE SYNONYM AI FOR AUTO_INSURER;
CREATE SYNONYM H_SA FOR HAS_SA;
CREATE SYNONYM POL FOR POLICY;
```

Figura 13: *DDL Statements, Creation of synonyms.*

Figure 14 below shows the automatically generated relational model in SQL Developer.

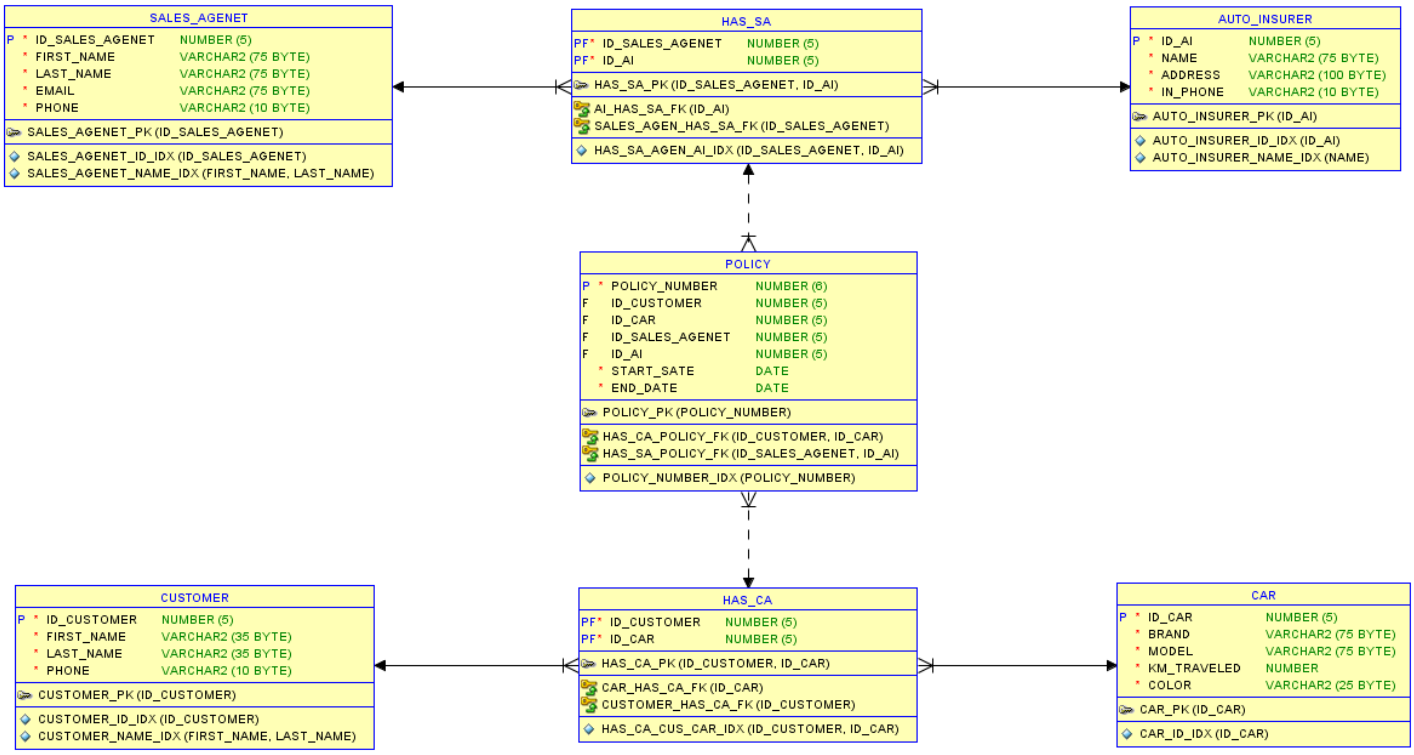


Figura 14: *Relational model.*

4. Pre-assessment

In this section you will find the Pre-assessment

Criteria to be evaluate	Does it comply?	(%)
COMPLIES WITH THE REQUESTED FUNCTIONALITY	YES	
HAS THE CORRECT INDENTATION	YES	
HAS AN EASY WAY TO ACCESS THE PROVIDED FILES	YES	
HAS A REPORT WITH IDC FORMAT	YES	
REPORT INFORMATION IS FREE OF SPELLING ERRORS	YES	
DELIVERED IN TIME AND FORM	YES	
IS FULLY COMPLETED (SPECIFY THE PERCENTAGE COMPLETED)	YES	100 %

5. Conclusion

The Oracle DDL language is transcendental in the handling of SQL statements at the level of both administrator and database programmer, since it allows the definition of database schemes regardless of the platform used to generate it.

This practice was very important for me since it helped me to remember and reinforce my knowledge in the use of DDL statements.