Object-relational database

Erick Gonzalez Parada 178145
Antonio Gutiérrez Blanco 177442
Andre Francois Duhamel Gutierrez 177315
Emiliano Ruiz Plancarte 177478

February 7, 2025

1 Objective

To design, implement, and evaluate an object-relational database system by defining complex object types, creating relational tables with nested collections, and establishing relationships between objects. The goal is to demonstrate the effective use of Object Query Language (OQL) for managing object-relational data structures, inserting sample data, and performing queries to validate the schema's functionality and integrity.

2 Introduction

There is an EXTRA section at the end
please don't forget

Object Query Language (OQL) is a standardized query language designed for querying and manipulating object-oriented databases. Unlike traditional SQL, which is tailored for relational databases, OQL is specifically built to handle complex object-oriented data structures, such as inheritance, polymorphism, and nested collections. OQL allows developers to interact with object-relational databases in a way that aligns with the principles of object-oriented programming, making it a powerful tool for modern database systems[2].
OQL revolves around its ability to query objects and their relationships directly, rather than relying on flat, tabular structures. This is particularly useful

in scenarios where data is represented as objects with attributes, methods, and relationships to other objects[1].

3 Literature Review

Traditional SQL (Structured Query Language) is designed for relational databases, where data is stored in flat, tabular structures. The philosophy of SQL revolves around the use of **joins** to combine data from multiple tables, enabling users to retrieve related information. For example, to retrieve data about an employee and their department, SQL requires joining the `Employee` and `Department` tables using a common key. While this approach is effective for relational data, it becomes cumbersome and inefficient when dealing with complex, hierarchical, or object-oriented data structures[3].

In contrast, Object Query Language (OQL) is specifically designed for object-oriented databases, where data is represented as objects with attributes, methods, and relationships. OQL eliminates the need for explicit joins by allowing developers to **directly traverse object relationships**. For instance, in an object-relational database, an `Employee` object might have a direct reference to a `Department` object. With OQL, querying the department of an employee is as simple as navigating the object hierarchy (e.g., `employee.department`), without the need for complex joins or intermediate tables[1].

While SQL can simulate some object-oriented behavior—such as using foreign keys and nested queries to represent relationships—it lacks the native support for object-oriented principles like **inheritance**, **polymorphism**, and **nested collections**. For example, simulating inheritance in SQL requires creating multiple tables and using complex joins to represent parent-child relationships. Similarly, representing nested collections (e.g., a list of phone numbers for an employee) often involves creating additional tables and performing multiple joins, which can lead to inefficient queries and increased complexity[2].

OQL, on the other hand, natively supports these object-oriented features. It allows developers to define **complex object types**, **inheritance hierarchies**, and **nested collections** directly in the database schema. This makes OQL more intuitive and efficient for querying object-oriented data, as it aligns with the natural structure of the data. For example, querying a nested collection of phone numbers in OQL is straightforward (e.g., `employee.phones`), whereas in SQL, it requires joining multiple tables and aggregating results[1].

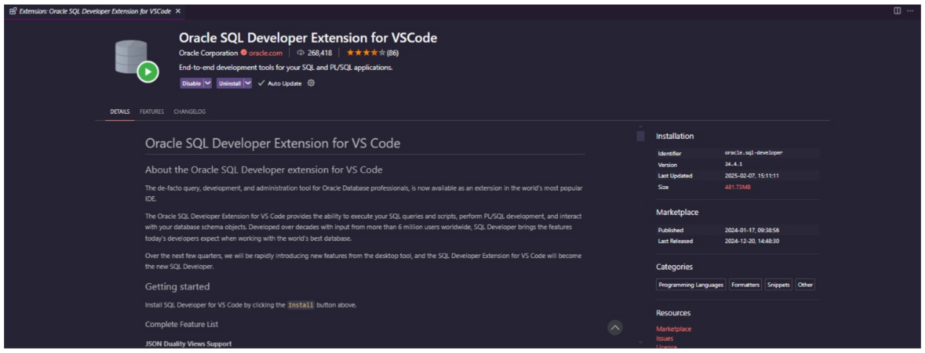
In summary, while SQL remains a powerful tool for relational data, its reliance on joins and lack of native support for object-oriented principles make it less suitable for modern, object-oriented applications. OQL addresses these limitations by providing a query language that is inherently designed for object-relational databases, enabling more efficient and intuitive data retrieval and manipulation.

4 Methodology

To implement and evaluate the object-relational database system, we used **Oracle Database Express Edition (XE)** as the primary database management system. The general solution involved the following steps:

1. **Setting Up Oracle Database Express**: - Oracle Database XE was downloaded and installed on a local machine. During installation, the default settings were used, and a system password was set for the administrative user (`SYS`). - The database was configured to allow remote connections, and the necessary ports (e.g., 1521 for Oracle) were opened to enable access from external tools.

2. **Configuring Visual Studio Code (VSCode)**: - The **Oracle Developer Tools for VSCode** extension was installed to facilitate database connectivity and query execution. - A connection profile was created in VSCode, specifying the host (e.g., `localhost`), port (e.g., `1521`), service name (e.g., `XE`), and credentials (e.g., `SYS` and the system password).



Oracle SQL Developer Extension for VSCode
Oracle Corporation oracle.com | 268,418 | ★★★★☆ (86)
End-to-end development tools for your SQL and PL/SQL applications.

Oracle SQL Developer Extension for VS Code

About the Oracle SQL Developer extension for VS Code

The de-facto query, development, and administration tool for Oracle Database professionals, is now available as an extension in the world's most popular IDE.

The Oracle SQL Developer Extension for VS Code provides the ability to execute your SQL queries and scripts, perform PL/SQL development, and interact with your database schema objects. Developed over decades with input from more than 6 million users worldwide, SQL Developer brings the features today's developers expect when working with the world's best database.

Over the next few quarters, we will be rapidly introducing new features from the desktop tool, and the SQL Developer Extension for VS Code will become the new SQL Developer.

Getting started

Install SQL Developer for VS Code by clicking the `Install` button above.

Complete Feature List

JSON Duality Views Support

Installation

Identifier	oracle.sql-developer
Version	24.4.1
Last Updated	2025-02-07, 15:11:11
Size	481.73MB

Marketplace

Published	2024-01-17, 09:38:56
Last Released	2024-12-20, 14:48:30

Categories

Programming Languages | Formatters | Snippets | Other

Resources

Marketplace
Issues

Figure 1: VSCode extension

3. **Writing and Executing OQL Scripts**: - OQL scripts were written to define complex object types, create relational tables with nested collections, and establish relationships between objects. These scripts were saved as `.sql` files for execution. - The scripts included: - **Object Type Definitions**: Creating object types such as `employee_t`, `worker_t`, and `manager_t` with attributes and methods. - **Table Creation**: Defining tables with nested collections (e.g., `telephone_t` as a nested table of phone numbers). - **Data Insertion**: Inserting sample data into the tables to populate the database. - **Query Execution**: Writing and executing queries to retrieve and validate data from the object-relational schema.

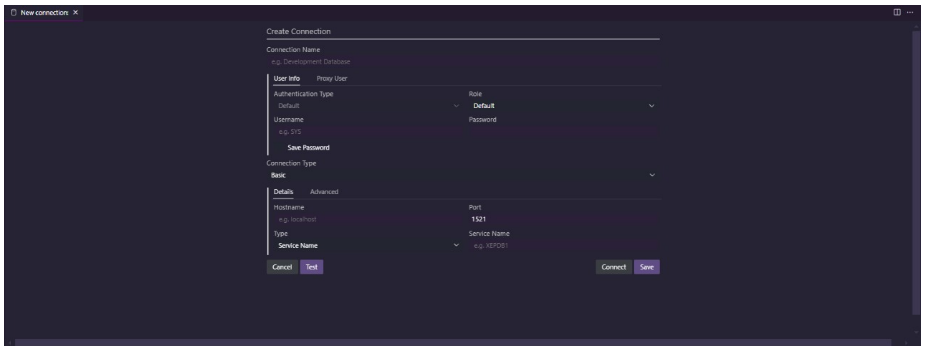
4. **Challenges Faced**: - One of the major challenges was the **lack of comprehensive online documentation** for OQL, especially in the context of Oracle Database. This made it difficult to troubleshoot issues and find examples of advanced OQL features. - Additionally, configuring the Oracle extension in

VSCode required trial and error, as some settings (e.g., connection parameters) were not well-documented.

5. **Validation and Testing**: - After executing the OQL scripts, the database schema and data were validated by running queries to ensure that object relationships, nested collections, and inheritance hierarchies were functioning as expected. - The results were compared with the expected outcomes to verify the correctness and efficiency of the implemented schema.

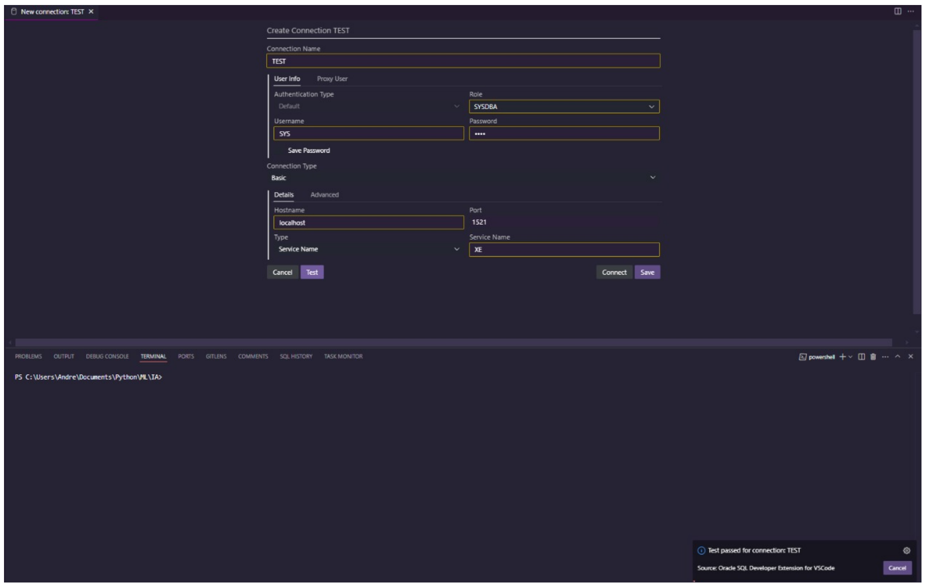
5 Result Analysis

The following figures will show us the whole work done for our solution and how the oql compiled successfully:



Connection Name: e.g. Development Database
Authentication Type: Default
Role: Default
Username: e.g. SYS
Password:
Connection Type: Basic
Hostname: e.g. localhost
Port: 1521
Type: Service Name
Service Name: e.g. XEPDB1

Figure 2: Create connection



Create Connection TEST

Connection Name: TEST

User Info | Proxy User

Authentication Type: Default

Role: SYSDBA

Username: SYS

Password: ••••

Save Password

Connection Type: Basic

Details | Advanced

Hostname: localhost

Port: 1521

Type: Service Name

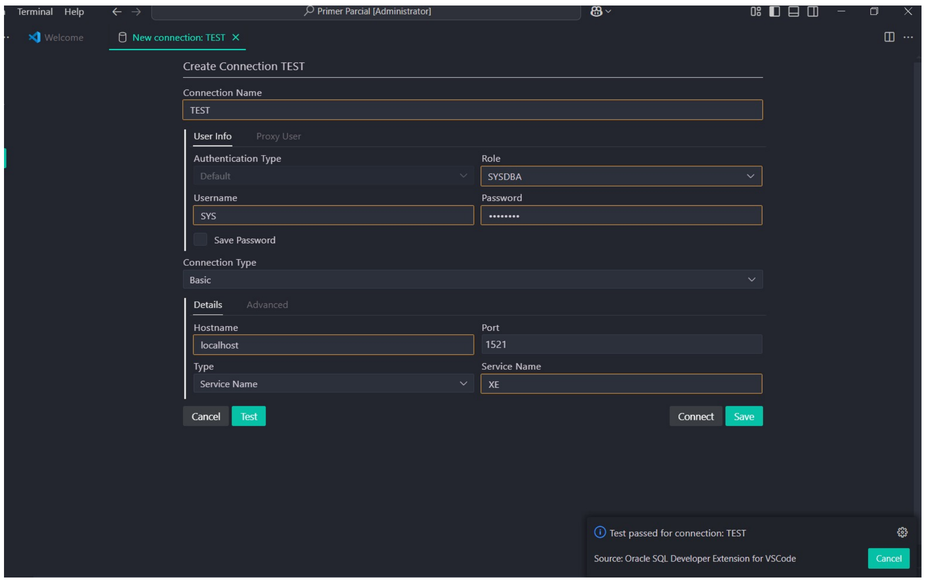
Service Name: XE

```
PS C:\Users\Andre\Documents\Python\ML\IA>
```

Test passed for connection: TEST

Source: Oracle SQL Developer Extension for VSCode

Figure 3: Setting connection



Create Connection TEST

Connection Name: TEST

User Info | Proxy User

Authentication Type: Default

Role: SYSDBA

Username: SYS

Password: ••••••••

Save Password

Connection Type: Basic

Details | Advanced

Hostname: localhost

Port: 1521

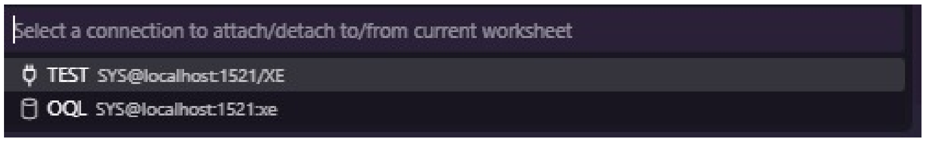
Type: Service Name

Service Name: XE

Test passed for connection: TEST

Source: Oracle SQL Developer Extension for VSCode

Figure 4: Setting connection

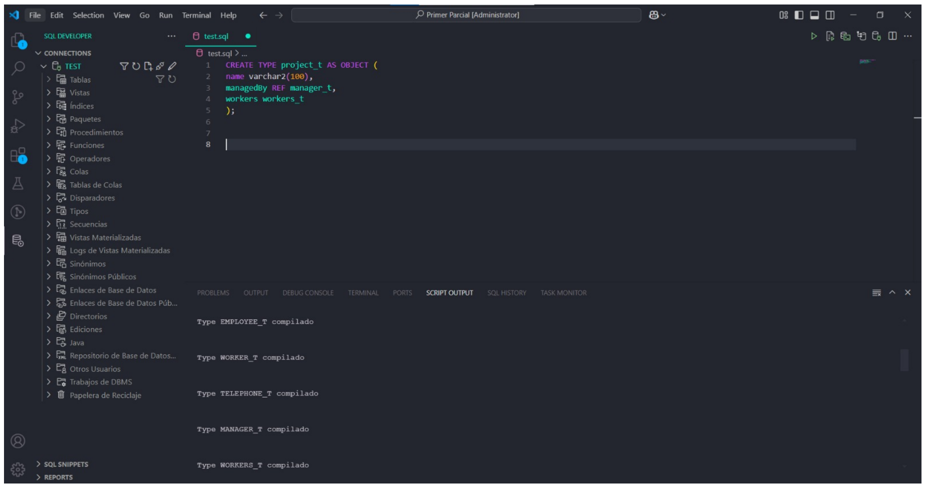


Select a connection to attach/detach to/from current worksheet

TEST SYS@localhost:1521/XE

OQL SYS@localhost:1521:xe

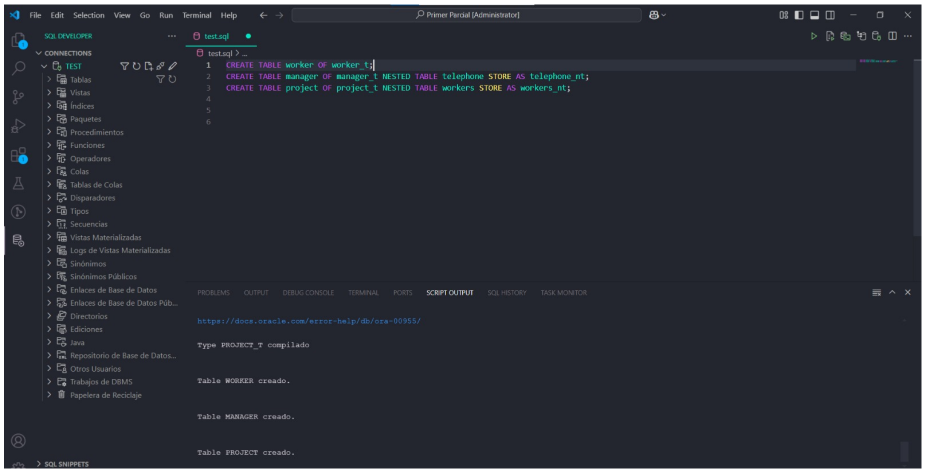
Figure 5: Connection selection



```sql
CREATE TYPE project_t AS OBJECT (
name varchar2(100),
managedBy REF manager_t,
workers workers_t
);
```

```
Type EMPLOYEE_T compilado

Type WORKER_T compilado

Type TELEPHONE_T compilado

Type MANAGER_T compilado

Type WORKERS_T compilado
```

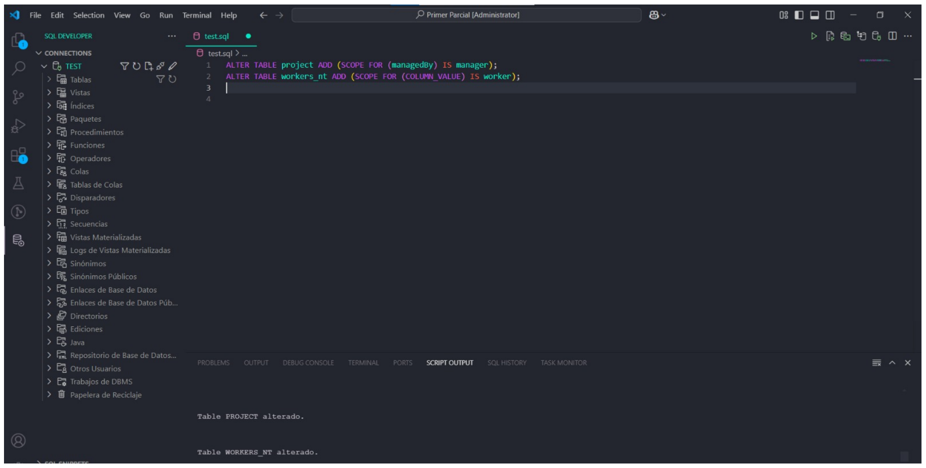
Figure 6: Create query



```sql
CREATE TABLE worker OF worker_t;
CREATE TABLE manager OF manager_t NESTED TABLE telephone STORE AS telephone_nt;
CREATE TABLE project OF project_t NESTED TABLE workers STORE AS workers_nt;
```

```
https://docs.oracle.com/error-help/db/ora-00955/

Type PROJECT_T compilado


Table WORKER creado.


Table MANAGER creado.


Table PROJECT creado.
```

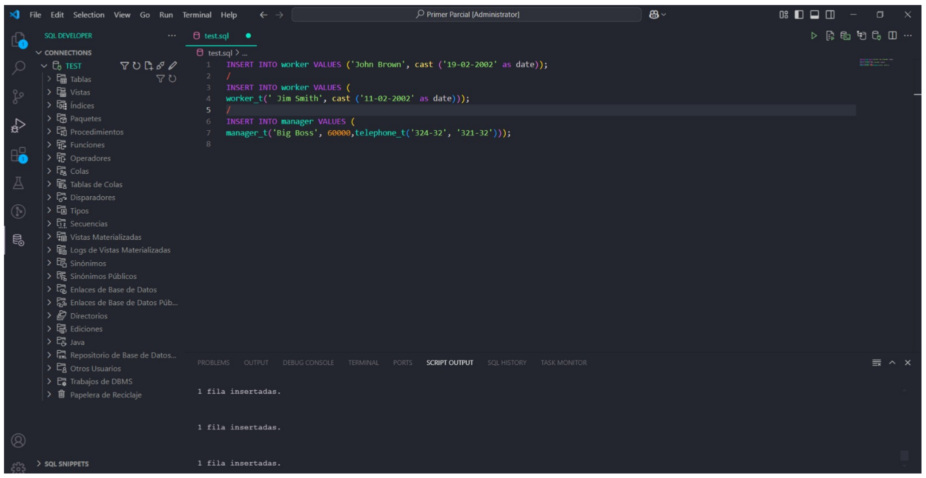
Figure 7: Create query



```sql
ALTER TABLE project ADD (SCOPE FOR (managedBy) IS manager);
ALTER TABLE workers_nt ADD (SCOPE FOR (COLUMN_VALUE) IS worker);
```

```
Table PROJECT alterado.


Table WORKERS_NT alterado.
```

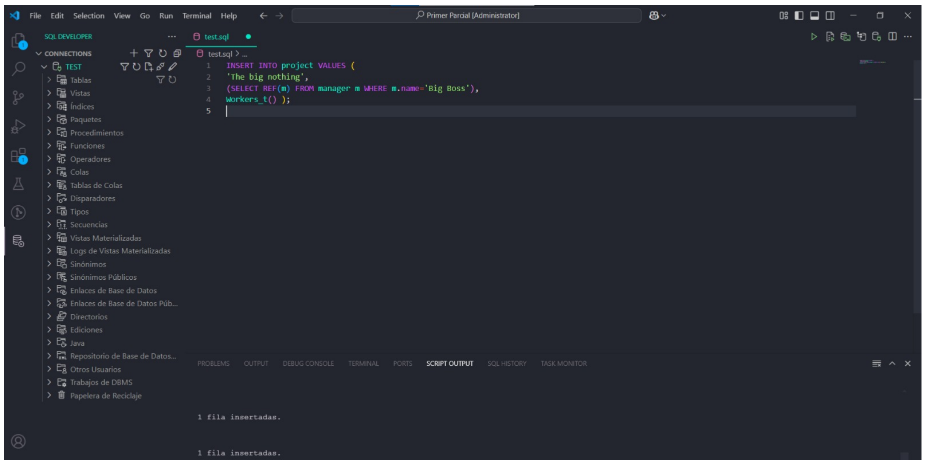
Figure 8: Alter query



```sql
INSERT INTO worker VALUES ('John Brown', cast ('19-02-2002' as date));
/
INSERT INTO worker VALUES (
worker_t(' Jim Smith', cast ('11-02-2002' as date)));
/
INSERT INTO manager VALUES (
manager_t('Big Boss', 60000,telephone_t('324-32', '321-32')));
```

```
1 fila insertadas.


1 fila insertadas.


1 fila insertadas.
```

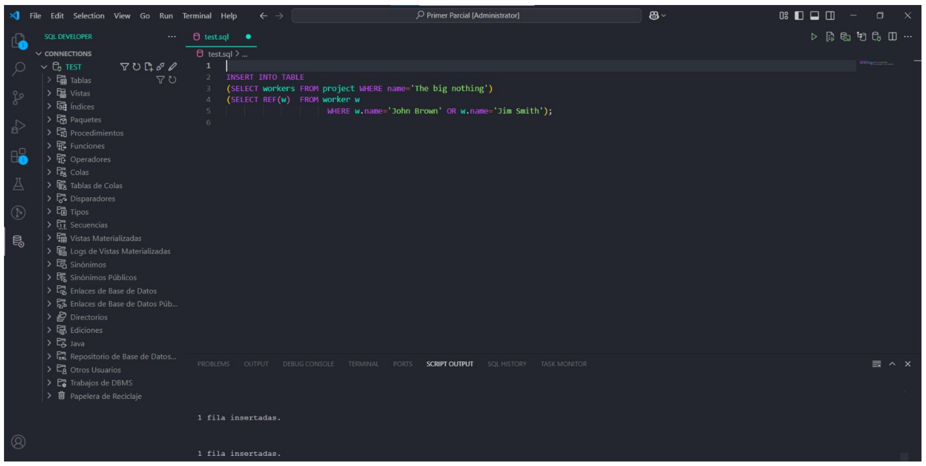
Figure 9: Insert query



```sql
INSERT INTO project VALUES (
'The big nothing',
(SELECT REF(m) FROM manager m WHERE m.name='Big Boss'),
Workers_t() );
```

```
1 fila insertadas.


1 fila insertadas.
```

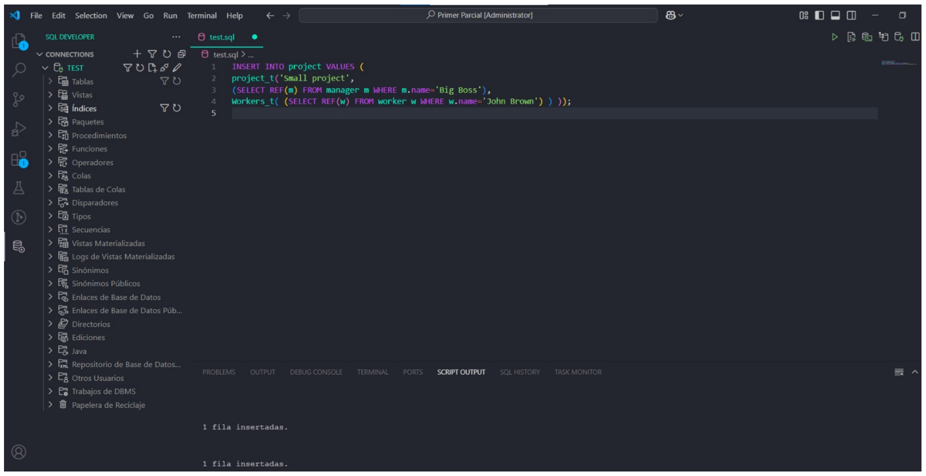
Figure 10: Insert query



```sql

INSERT INTO TABLE
(SELECT workers FROM project WHERE name='The big nothing')
(SELECT REF(w)  FROM worker w
                    WHERE w.name='John Brown' OR w.name='Jim Smith');

```

```
1 fila insertadas.

1 fila insertadas.
```

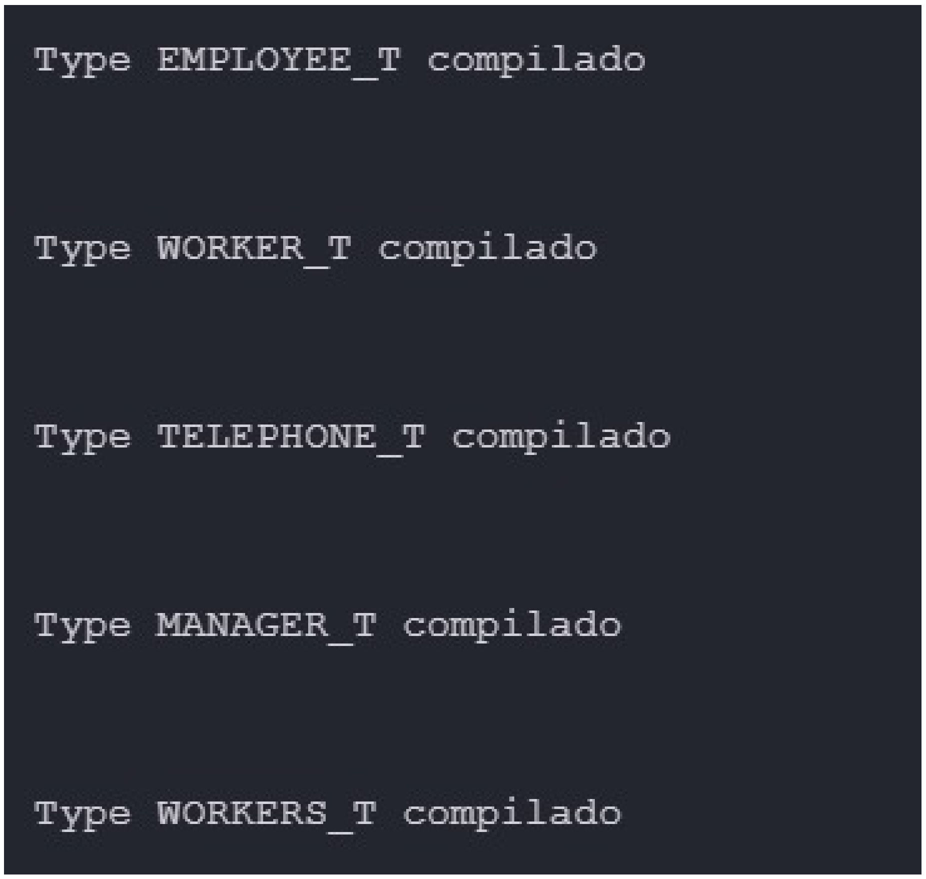
Figure 11: Insert query



```sql
INSERT INTO project VALUES (
project_t('Small project',
(SELECT REF(m) FROM manager m WHERE m.name='Big Boss'),
Workers_t( (SELECT REF(w) FROM worker w WHERE w.name='John Brown') ) ));

```

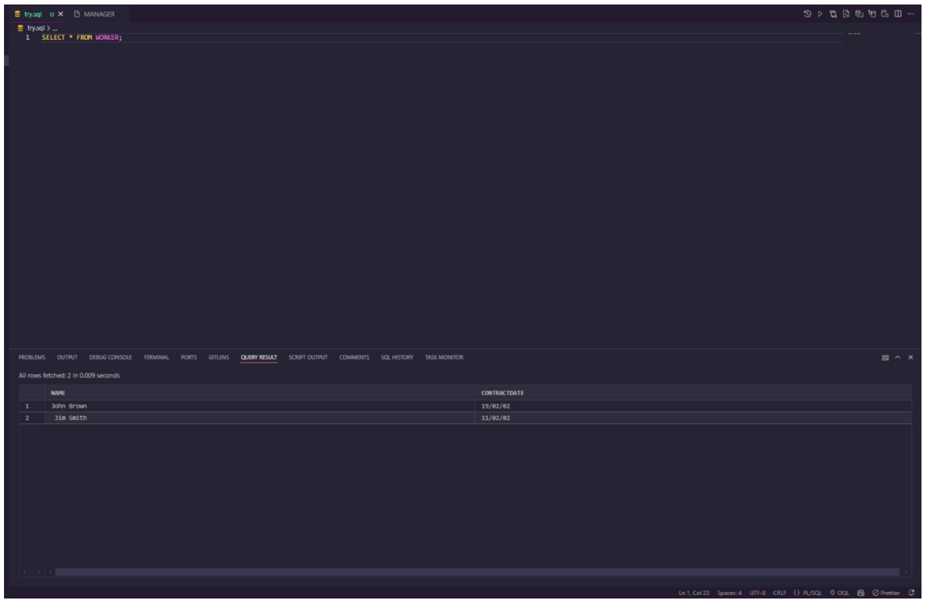
```
1 fila insertadas.

1 fila insertadas.
```

Figure 12: Insert query



```
Type EMPLOYEE_T compilado


Type WORKER_T compilado


Type TELEPHONE_T compilado


Type MANAGER_T compilado


Type WORKERS_T compilado
```

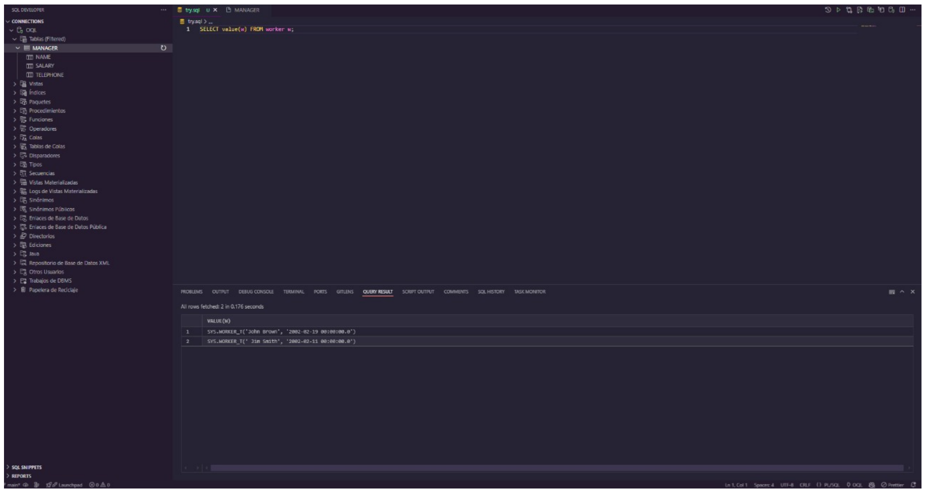
Figure 13: compilation successfully



```sql
SELECT * FROM WORKER;
```

```
All rows fetched: 2 in 0.009 seconds

     NAME            CONTRACTDATE
1    John Brown      19/02/02
2     Jim Smith      11/02/02
```

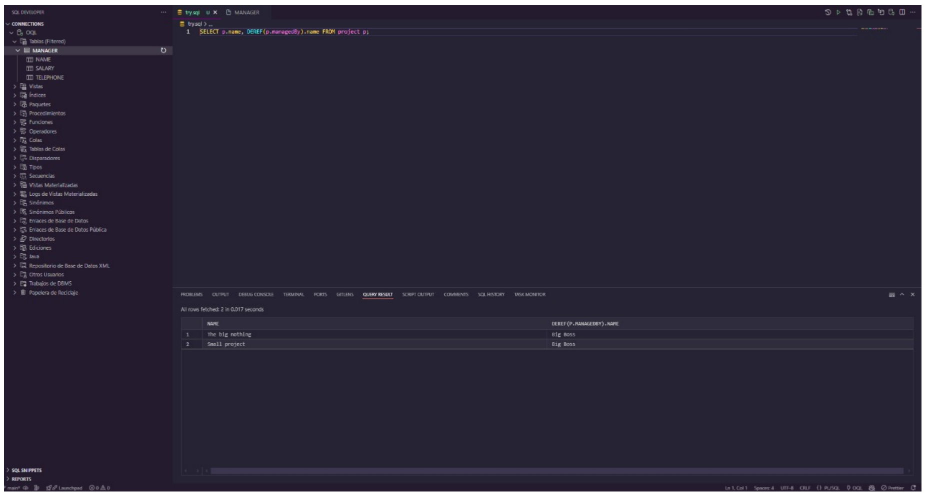
Figure 14: Select query



```sql
SELECT value(w) FROM worker w;
```

```
All rows fetched: 2 in 0.176 seconds

     VALUE(W)
1    SYS.WORKER_T('John Brown', '2002-02-19 00:00:00.0')
2    SYS.WORKER_T(' Jim Smith', '2002-02-11 00:00:00.0')
```

Figure 15: Select query

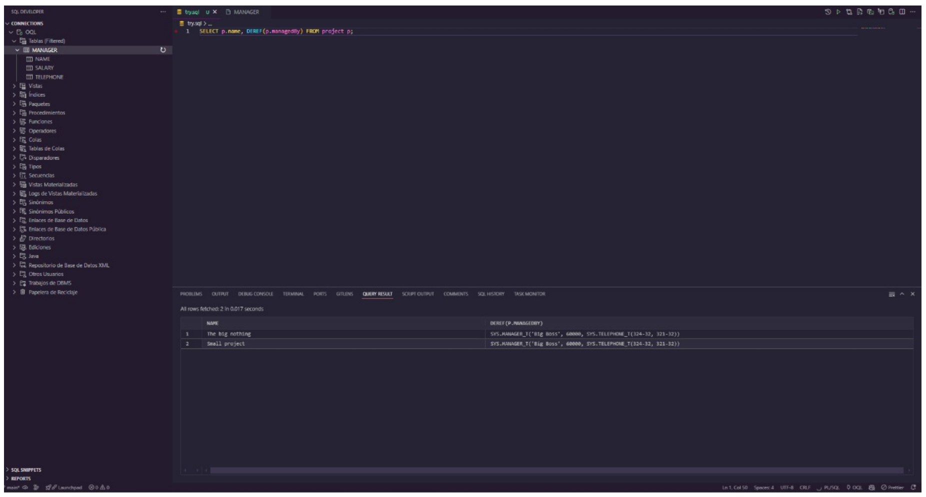


```sql
SELECT p.name, DEREF(p.managedBy).name FROM project p;
```

All rows fetched: 2 in 0.017 seconds

	NAME	DEREF(P.MANAGEDBY).NAME
1	The big nothing	Big Boss
2	Small project	Big Boss

Figure 16: Select query

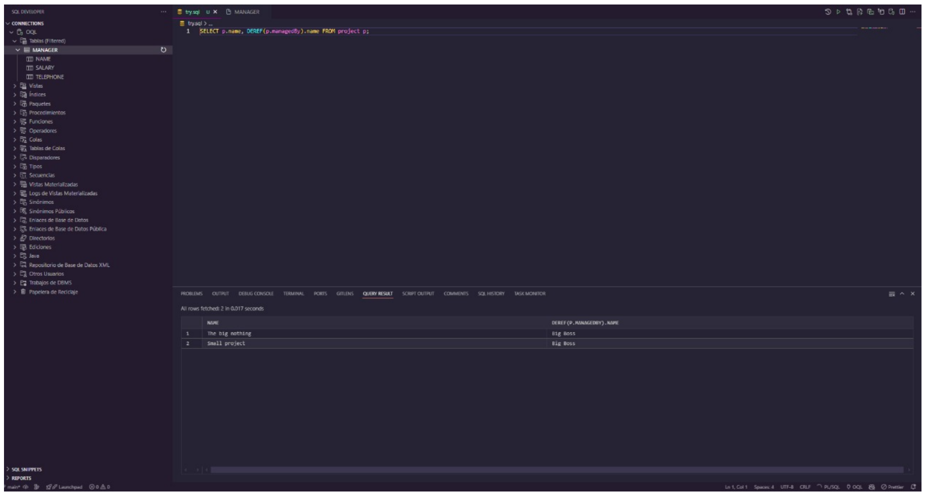


```sql
SELECT p.name, DEREF(p.managedBy) FROM project p;
```

All rows fetched: 2 in 0.017 seconds

	NAME	DEREF(P.MANAGEDBY)
1	The big nothing	SYS.MANAGER_T('Big Boss', 60000, SYS.TELEPHONE_T(324-32, 321-32))
2	Small project	SYS.MANAGER_T('Big Boss', 60000, SYS.TELEPHONE_T(324-32, 321-32))

Figure 17: Select query

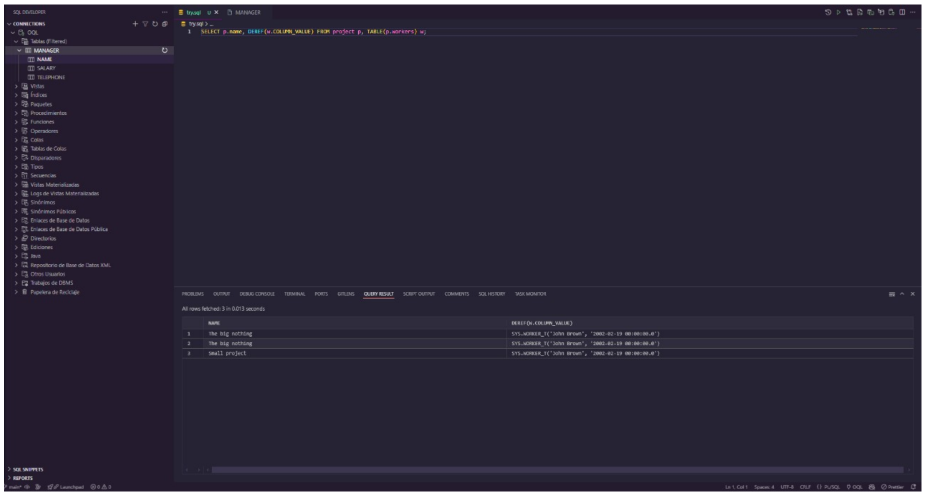


```sql
SELECT p.name, DEREF(p.managedBy).name FROM project p;
```

All rows fetched: 2 in 0.017 seconds

	NAME	DEREF(P.MANAGEDBY).NAME
1	The big nothing	Big Boss
2	Small project	Big Boss

Figure 18: Select query



```sql
SELECT p.name, DEREF(w.COLUMN_VALUE) FROM project p, TABLE(p.workers) w;
```

All rows fetched: 3 in 0.013 seconds

	NAME	DEREF(W.COLUMN_VALUE)
1	The big nothing	SYS.WORKER_T('John Brown', '2002-02-19 00:00:00.0')
2	The big nothing	SYS.WORKER_T('John Brown', '2002-02-19 00:00:00.0')
3	Small project	SYS.WORKER_T('John Brown', '2002-02-19 00:00:00.0')

Figure 19: Select query

6 Conclusions

This work demonstrated the effectiveness of Object Query Language (OQL) in managing object-relational databases. **However we as a team struggled a ton to get this tools working due to the lack of documentation that**

is online. By defining complex object types, nested collections, and relationships, we showed that OQL simplifies querying object-oriented data compared to SQL, which relies on complex joins. OQL's native support for object-oriented principles, such as inheritance and polymorphism, makes it a superior choice for modern applications. This highlights OQL's potential for advancing database technologies in object-oriented contexts.

7 EXTRA

References

[1] OQL. (n.d.). Mendix Documentation. Retrieved February 7, 2025, from https://docs.mendix.com/refguide/oql/

[2] Tivoli Network Manager IP edition 4.2.0. (2025, January 21). Ibm.com. https://www.ibm.com/docs/en/networkmanager/4.2.0?topic=reference-object-query-language

[3] What is SQL. (n.d.). W3schools.com. Retrieved February 7, 2025, from https://www.w3schools.com/whatis/whatis_sql.asp