

Universidad Autonoma del Estado de Hidalgo

Instituto de Ciencias Basicas e Ingenieria

Algebra relacional y SQL

Magaly Hernandez Reyes

6 Semestre Grupo 2

Bases de Datos Distribuidas

0.1. introducción

El álgebra relacional es un sistema formal para manipular y consultar bases de datos relacionales. Es el fundamento teórico de las operaciones que se realizan en bases de datos, como las que usamos con SQL. El álgebra relacional proporciona los fundamentos teóricos que SQL implementa. Mientras que el lenguaje relacional es más formal y matemático.

0.2. Marco teórico

El lenguaje relacional es un lenguaje de consulta procedimental que se utiliza para solicitar información de la base de datos. El usuario a través del álgebra relacional indica al sistema que lleve a cabo una serie de operaciones en la base de datos para calcular el resultado deseado.

El álgebra relacional nos brinda ayuda en las consultas SQL que escribimos no solo para que sean válidas, sino que también se puedan ejecutar de manera eficiente y correcta.

0.3. Desarrollo

Ejercicios practicos

El conjunto de ejercicios está basado en las tablas Employee y Reward. Se pide que para cada ejercicio se incluya una captura de pantalla con la sentencia SQL de solución y del resultado de su ejecución.

EMPLOYEE TABLE

Employee_id	First_name	Last_name	Salary	Joining_date	Departament
1	Bob	Kinto	1000000	2019-01-20	Finance
2	Jerry	Kansxo	6000000	2019-01-15	IT
3	Philip	Jose	8900000	2019-02-05	Banking
4	Jonh	Abraham	2000000	2019-02-25	Insurance
5	Michael	Mathew	2200000	2019-02-28	Finance
6	Alex	Chreketo	4000000	2019-05-10	IT
7	Yohan	Soso	1230000	2019-06-20	Banking

REWARD TABLE

Employee_ref_id	Date_reward	Amount
1	2019-05-11	1000
2	2019-02-15	5000
3	2019-04-22	2000
4	2019-06-20	8000

EJERCICIOS

1. Escribe la sintaxis para crear la tabla “Employee”.

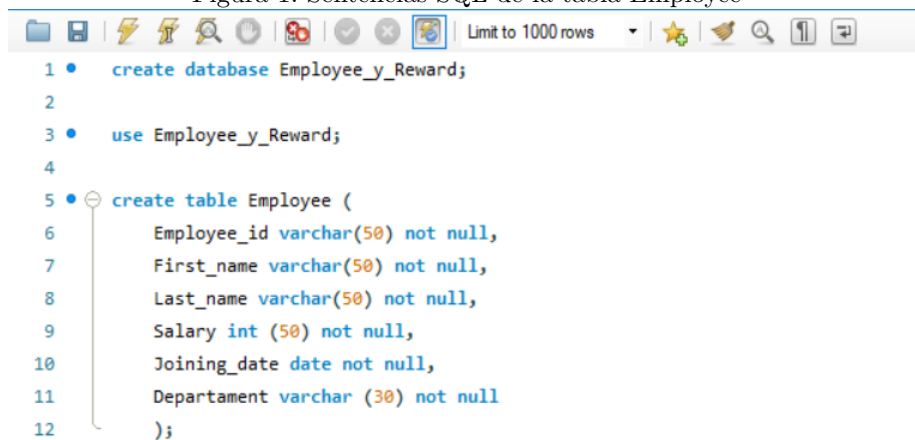
Algebra relacional:

π Employee_id, First_name, Last_name, Salary, Joining_date, Departament(Employee)

Sentencia SQL:

```
create table Employee (  
    Employee_id varchar(50) not null,  
    First_name varchar(50) not null,  
    Last_name varchar(50) not null,  
    Salary int (50) not null,  
    Joining_date date not null,  
    Departament varchar (30) not null  
);
```

Figura 1: Sentencias SQL de la tabla Employee



2. Escribe la sintaxis para insertar 7 registros (de la imagen) a la tabla “Employee”.

Algebra relacional:

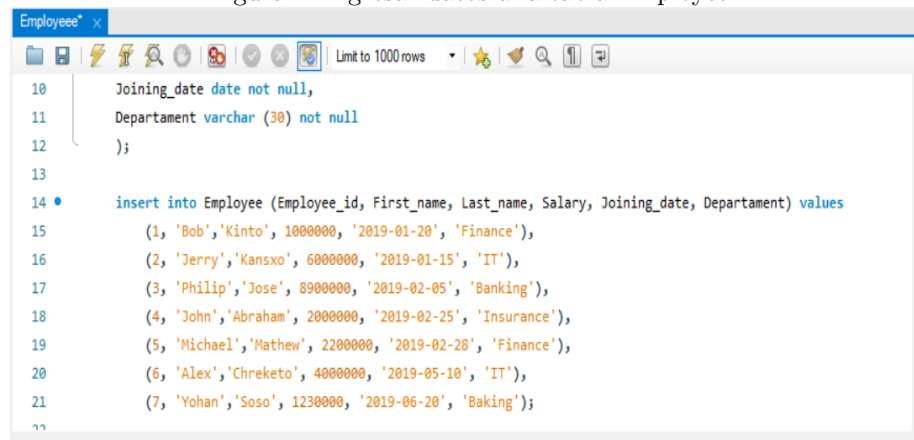
```
Employee ← Employee ∪ {1, 'Bob', 'Kinto', 1000000, '2019-01-20', 'Finance'}  
Employee ← Employee ∪ {2, 'Jerry', 'Kansxo', 6000000, '2019-01-15', 'IT'}  
Employee ← Employee ∪ {3, 'Philip', 'Jose', 8900000, '2019-02-05', 'Banking'}  
Employee ← Employee ∪ {4, 'John', 'Abraham', 2000000, '2019-02-25', 'Insurance'}  
Employee ← Employee ∪ {5, 'Michael', 'Mathew', 2200000, '2019-02-28', 'Finance'}  
Employee ← Employee ∪ {6, 'Alex', 'Chreketo', 4000000, '2019-05-10', 'IT'}  
Employee ← Employee ∪ {7, 'Yohan', 'Soso', 1230000, '2019-06-20', 'Baking'}
```

Sentencias SQL:

```
insert into Employee (Employee_id, First_name, Last_name,  
Salary, Joining_date, Departament) values  
(1, 'Bob', 'Kinto', 1000000, '2019-01-20', 'Finance'),  
(2, 'Jerry', 'Kansxo', 6000000, '2019-01-15', 'IT'),  
(3, 'Philip', 'Jose', 8900000, '2019-02-05', 'Banking'),  
(4, 'John', 'Abraham', 2000000, '2019-02-25', 'Insurance'),  
(5, 'Michael', 'Mathew', 2200000, '2019-02-28', 'Finance'),  
(6, 'Alex', 'Chreketo', 4000000, '2019-05-10', 'IT'),  
(7, 'Yohan', 'Soso', 1230000, '2019-06-20', 'Baking');
```

```
Select * from Employee;
```

Figura 2: Ingresar datos a la tabla Employee



3. Escribe la sintaxis para crear la tabla “Reward”.

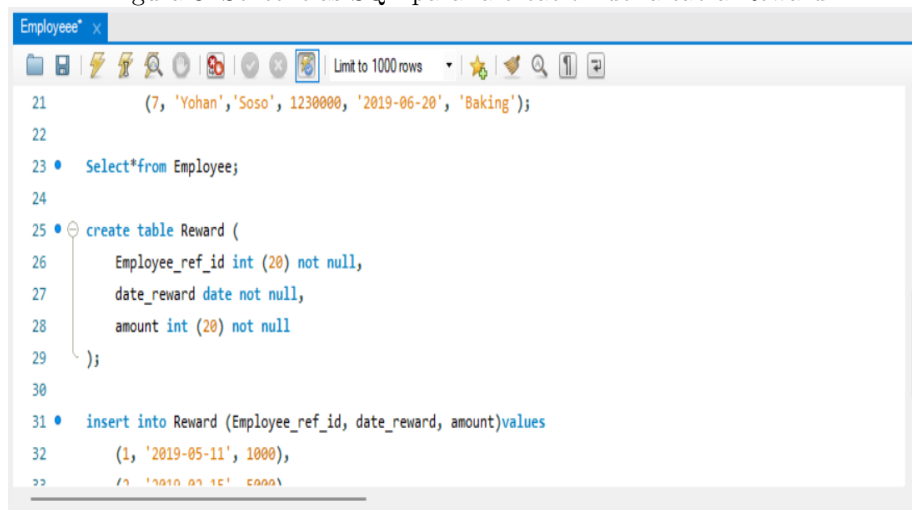
Algebra relacional:

π Employee_ref_id, date_reward, amount(Reward)

Sentencia SQL:

```
create table Reward (  
  Employee_ref_id int (20) not null,  
  date_reward date not null,  
  amount int (20) not null  
);
```

Figura 3: Sentencias SQL para la creacion de la tabla Reward



4. Escribe la sintaxis para insertar 4 registros (en la imagen) a la tabla “Reward”.

Algebra relacional:

$Reward \leftarrow Reward \cup \{1, '2019-05-11', 1000\}$

$Reward \leftarrow Reward \cup \{2, '2019-02-15', 5000\}$

$Reward \leftarrow Reward \cup \{3, '2019-04-22', 2000\}$

$Reward \leftarrow Reward \cup \{4, '2019-06-20', 8000\}$

Sentencias SQL:

```
insert into Reward (Employee_ref_id , date_reward ,  
amount) values  
(1 , '2019-05-11 ' , 1000) ,  
(2 , '2019-02-15 ' , 5000) ,  
(3 , '2019-04-22 ' , 2000) ,  
{ (4 , '2019-06-20 ' , 8000);
```



Figura 4: Sentencias ara llenar la tabla Reward

5. Obtener todos los empleados.

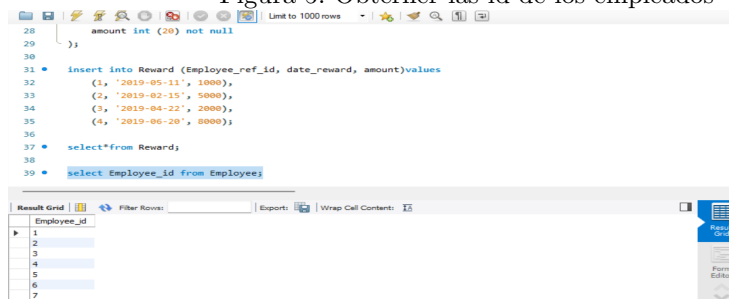
Algebra relacional:

$\prod Employee_id(Employee)$

Sentencia SQL:

```
select Employee_id from Employee;
```

Figura 5: Obtener las id de los empleados



6. Obtener el primer nombre y apellido de todos los empleados.

Algebra relacional:

$\pi_{First_name, Last_name}(Employee)$

Sentencias SQL:

```
select First_name , Last_name from Employee;
```

Figura 6: Tabla nombre y apellido

The screenshot shows a SQL IDE window titled "Employee* x". The script editor contains the following SQL commands:

```
30
31 • insert into Reward (Employee_ref_id, date_reward, amount)values
32     (1, '2019-05-11', 1000),
33     (2, '2019-02-15', 5000),
34     (3, '2019-04-22', 2000),
35     (4, '2019-06-20', 8000);
36
37 • select*from Reward;
38
39 • select Employee_id from Employee;
40
41 • select First_name, Last_name from Employee;
```

Below the script editor is a "Result Grid" section. It includes a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. The grid displays the results of the last query, showing a list of employees with their first and last names.

	First_name	Last_name
▶	Bob	Kinto
	Jerry	Kansxo
	Philip	Jose
	John	Abraham
	Michael	Mathew
	Alex	Chreketo
	Yohan	Soso

The bottom of the window shows a tab labeled "Employee 6 x".

7. Obtener todos los valores de la columna "First_name" usando el alias nombre de empleado.

Algebra relacional:

$\rho_{\text{Nombre_de_empleado}} / \text{First_name}(\pi_{\text{First_name}}(\text{Employee}))$

Sentencias SQL:

```
select First_name as Nombre_de_empleaado from Employee;
```

Figura 7: Obtener los nombres

The screenshot shows a SQL IDE window titled "Employee* x". The query editor contains the following SQL statements:

```
32      (1, '2019-05-11', 1000),
33      (2, '2019-02-15', 5000),
34      (3, '2019-04-22', 2000),
35      (4, '2019-06-20', 8000);
36
37 •   select * from Reward;
38
39 •   select Employee_id from Employee;
40
41 •   select First_name, Last_name from Employee;
42
43 •   select First_name as Nombre_de_empleaado from Employee;
```

The "Result Grid" tab is active, showing the results of the last query. The results are as follows:

Nombre_de_empleaado
Bob
Jerry
Philip
John
Michael
Alex
Yohan

8. Obtener todos los valores de la columna “Last_name” en minúsculas.

Algebra relacional:

No tiene transformacion

Sentencias SQL:

```
SELECT LOWER(Last_name) AS last_name_lowercase FROM Employee;
```

Figura 8: Tabla de nombre en minúscula

The screenshot displays a SQL IDE interface. The top pane shows a script editor with several SQL queries. The bottom pane shows the 'Result Grid' for the selected query, displaying a list of last names in lowercase.

SQL Script Editor:

```
34      (3, '2019-04-22', 2000),
35      (4, '2019-06-20', 8000);
36
37 • select*from Reward;
38
39 • select Employee_id from Employee;
40
41 • select First_name, Last_name from Employee;
42
43 • select First_name as Nombre_de_empleado from Employee;
44
45 • SELECT LOWER(Last_name) AS last_name_lowercase FROM Employee;
```

Result Grid:

last_name_lowercase
kinto
kansxo
jose
abraham
mathew
chreketo
soso

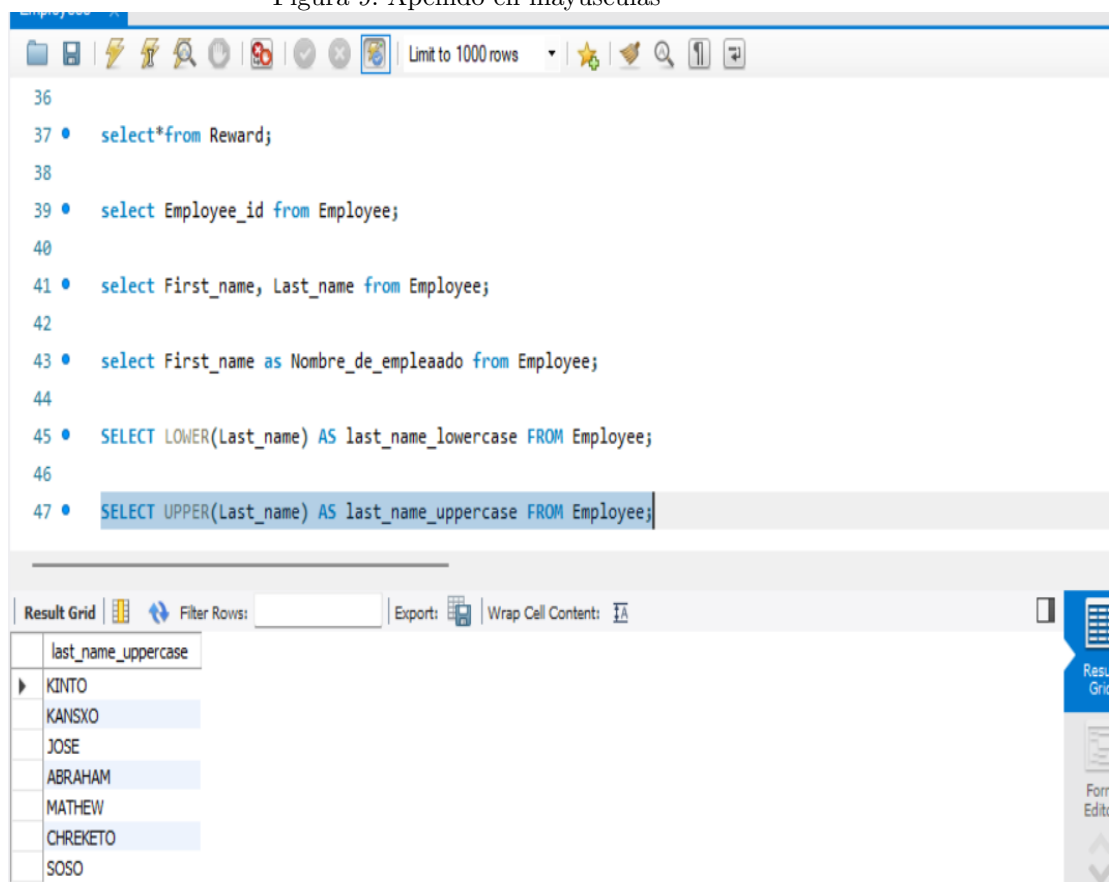
9. Obtener todos los valores de la columna “Last_name” en mayúsculas.

Algebra relacional:
No tiene transformacion

Sentencias SQL:

```
SELECT UPPER(Last_name) AS last_name_uppercase FROM Employee;
```

Figura 9: Apellido en mayusculas



10. Obtener los nombre únicos de la columna “Departament”.

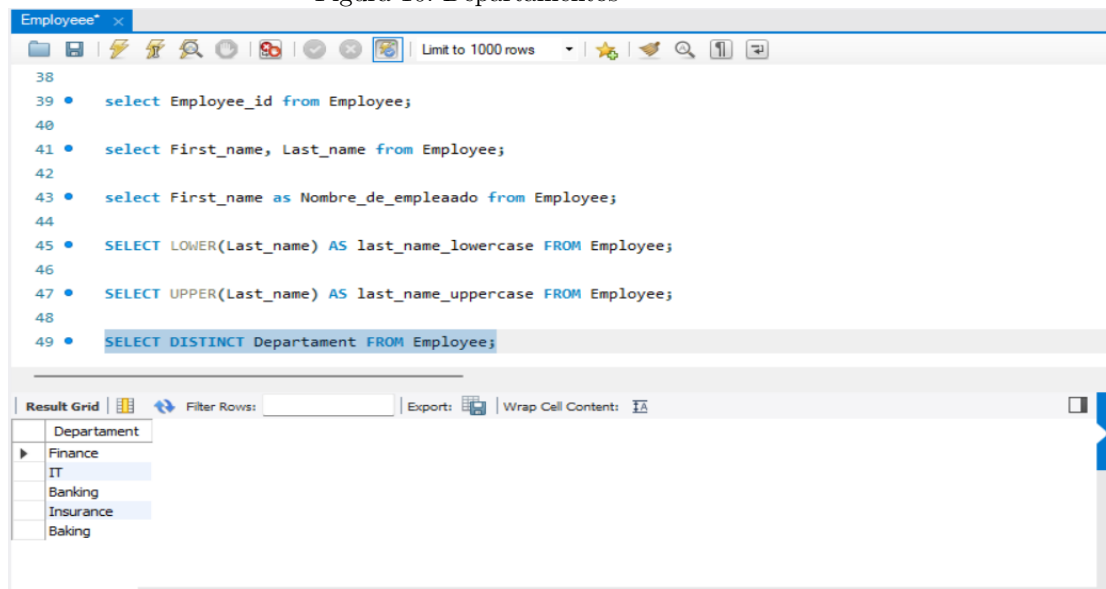
Algebra relacional:

$\pi_{\text{Departament}}(\text{Employee})$

Sentencia SQL:

SELECT DISTINCT Departament **FROM** Employee;

Figura 10: Departamentos



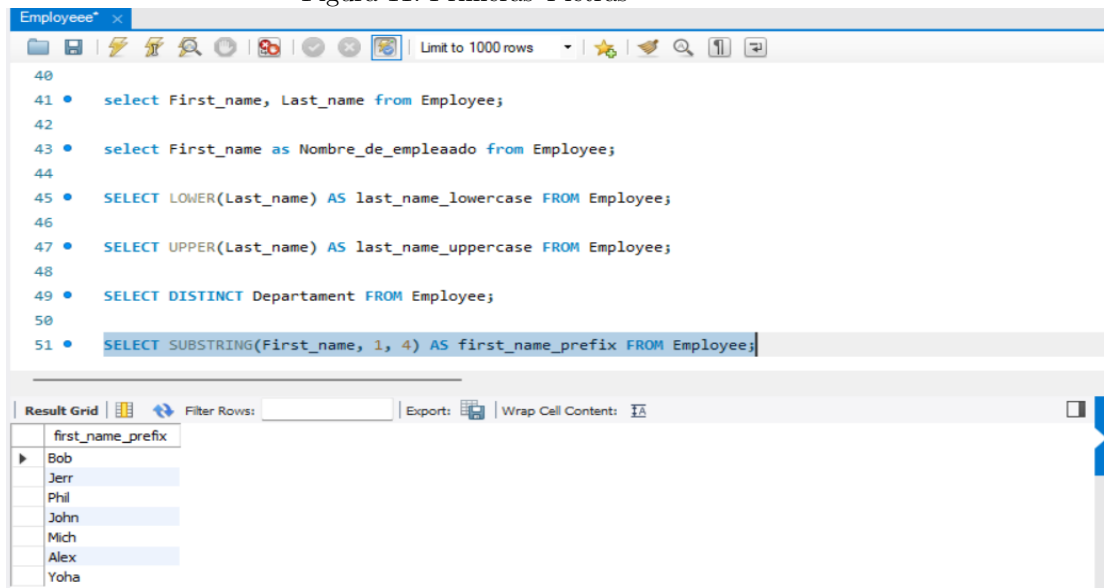
11. Obtener los primeros 4 caracteres de todos los valores de la columna “First_name”.

Algebra relacional:
No tiene Ecuacion

Sentencia SQL:

```
SELECT SUBSTRING(First_name , 1, 4) AS first_name_prefix  
FROM Employee;
```

Figura 11: Primeras 4 letras



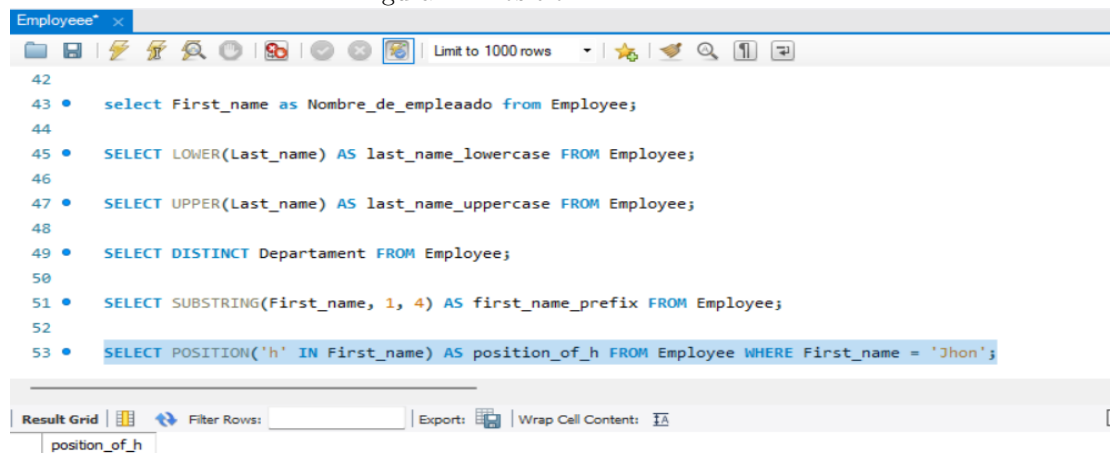
12. Obtener la posición de la letra “h” en el nombre del empleado con First_name = “Jhon”.

Algebra relacional:
No tiene Ecuación

Sentencia SQL:

```
SELECT POSITION('h' IN First_name) AS position_of_h  
FROM Employee WHERE First_name = 'Jhon';
```

Figura 12: Posición



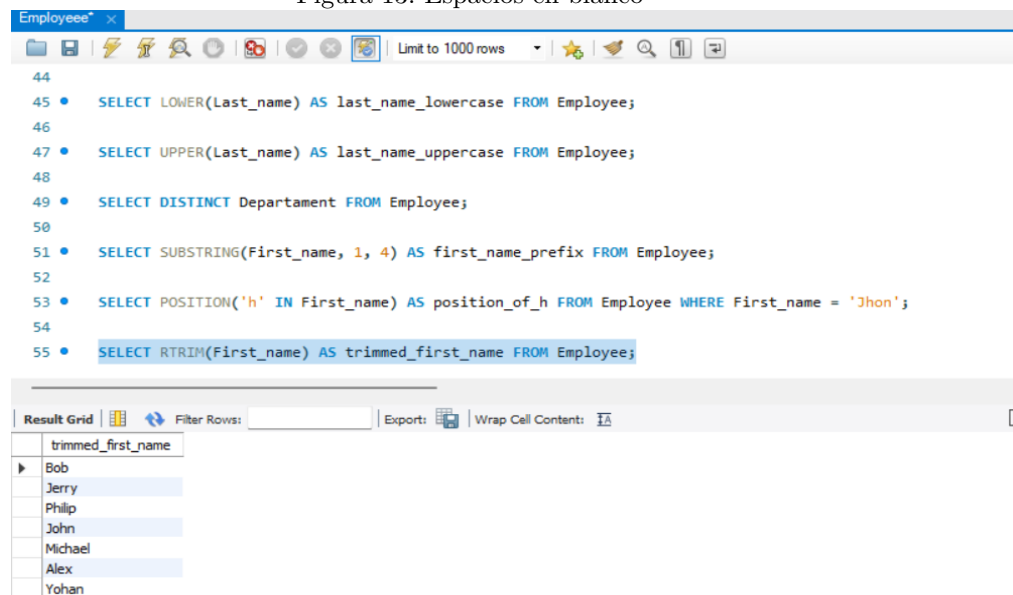
13. Obtener todos los valores de la columna “First_name” después de remover los espacios en blanco de la derecha.

Algebra relacional:
No tiene Ecuación

Sentencia SQL:

```
SELECT RTRIM(First_name) AS trimmed_first_name FROM Employee;
```

Figura 13: Espacios en blanco



The screenshot shows a SQL IDE window titled "Employee* x". The query editor contains the following SQL statements:

```
44
45 • SELECT LOWER>Last_name) AS last_name_lowercase FROM Employee;
46
47 • SELECT UPPER>Last_name) AS last_name_uppercase FROM Employee;
48
49 • SELECT DISTINCT Department FROM Employee;
50
51 • SELECT SUBSTRING(First_name, 1, 4) AS first_name_prefix FROM Employee;
52
53 • SELECT POSITION('h' IN First_name) AS position_of_h FROM Employee WHERE First_name = 'Jhon';
54
55 • SELECT RTRIM(First_name) AS trimmed_first_name FROM Employee;
```

The results pane shows the output of the last query, displaying a table with one column, "trimmed_first_name", and six rows of data:

trimmed_first_name
Bob
Jerry
Philip
John
Michael
Alex
Yohan

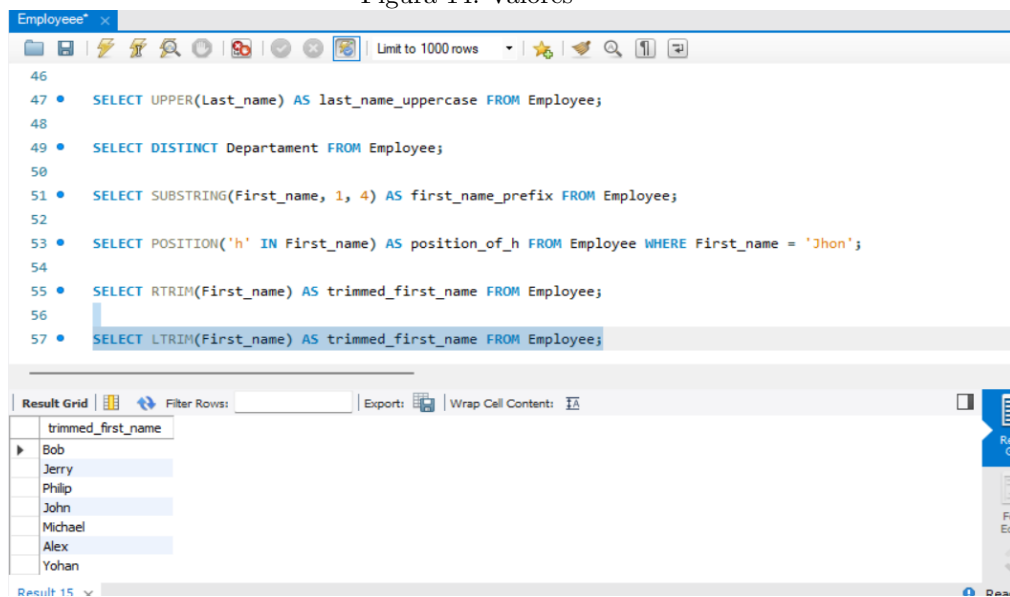
14. Obtener todos los valores de la columna “First_name” después de remover los espacios en blanco de la izquierda.

Algebra relacional:
No tiene Ecuación

Sentencia SQL:

```
SELECT LTRIM(First_name) AS trimmed_first_name FROM Employee;
```

Figura 14: Valores



0.4. Conclusión

Para concluir, podemos mencionar que las sentencias SQL nos brindan la capacidad de trabajar con datos de manera efectiva, asegurando su calidad, facilitando el análisis y la toma de decisiones, y asegurando que los sistemas que dependen de esos datos funcionen correctamente.