Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 1.3

Álgebra relacional y SQL

ALUMNO: Barón Salinas Mauricio Valentín
Dr. Eduardo Cornejo-Velázquez

1. Introducción

En el presente proyecto de investigación y prácticas acedémicas, se busca realizar ejercicio de conocimientos teóricos y prácticos en un entorno simulado para la mejora continua de las habilidades de diseño e implementación de sistemas de bases de datos.
El resultado propuesto aquí incentiva el ejercicio pleno de las habilidades duras y suaves para el desarrollo profesional del alumnado, así como de la investigación de la institución académica presente.

2. Marco teórico

Análisis de requerimientos

La empresa requiere de un sistema optimizado y a la medida para sus necesidades particulares, que le suministre herramientas para la optimización de su flujo de procesos y productos/servicios.
Tal sistema requiere de un diseño bien pensado que pueda abastecer cada uno de los objetivos que de antemano se reconocen con la ayuda del cliente, y a partir de su modelo de negocios, diagramas de visualización de estrategias FODA y PESTEL, así como de ciertos proyectos que se ciernen hacia el dominio del mercado nacional e internacional de la organización.

Modelo Entidad - Relación

2.2.1 Modelo

El modelo E/R se remonta al año 1970, cuando Peter Chen observó lo complejo que era diseñar una base de datos. Decidió crear un lenguaje común de símbolos que facilitara la comunicación (*Gomez, 2013*).

2.2.2 Entidad

Se denomina entidad a cualquier elemento sobre el cual se desea almacenar información. Aunque las entidades suelen corresponderse con los sustantivos, no todos los sustantivos merecen ser entidades que aparezcan en el modelo (*Gomez, 2013*).
Las entidades pueden ser fuertes o débiles:

- Una entidad **fuerte** es aquella que existe por sí sola.
- Una entidad **débil** es aquella que requiere que otra entidad exista antes que ella.

2.2.3 Interrelaciones

Son asociaciones entre entidades. En general se corresponden con los verbos. Se representan por medio de un rombo con el nombre dentro (*Gomez, 2013*).
Las interrelaciones pueden ser entre distintos conjuntos de entidades:

- Interrelación binaria: se da entre dos entidades.
- Interrelación ternaria: se da entre tres entidades que participan de forma simultánea en una asociación.
- Interrelación n-aria.
- Interrelaciones reflexivas: se dan entre una entidad y sí misma. El principal ejemplo se da en la sentencia “unos empleados son jefes de otros”.

Modelo relacional

2.3.1 Estructura

La estructura fundamental del modelo relacional es la relación, es decir una tabla bidimensional constituida por filas (tuplas) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la base de datos.

2.3.2 Tuplas

Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representan las propiedades de la entidad. Por ejemplo, si en la base de datos se tienen que representar personas, podrá definirse una relación llamada “Personas”, cuyos atributos describen las características de las personas. Cada tupla de la relación “Personas” representará una persona concreta.

2.3.3 Tablas y tuplas

Aunque una relación es más conocida como tabla, las tuplas como filas y los atributos como columnas, en este escrito usaremos la terminología original y de donde deriva el nombre del modelo. Las tuplas en una relación son un conjunto en el sentido matemático del término, es decir una colección no ordenada de elementos diferentes. Para distinguir una 55 tupla de otra, se recurre al concepto de "llave primaria", o sea un atributo o conjunto de atributos que permiten identificar unívocamente una tupla en una relación (en el ejemplo, el atributo RFC cumple con esta función).

2.3.4 Llaves primarias

Naturalmente, en una relación puede haber más combinaciones de atributos que permitan identificar unívocamente una tupla ("llaves candidatas"), pero entre éstas se elegirá una sola para utilizar como llave primaria. Los atributos de la llave primaria no pueden asumir el valor nulo (que significa un valor no determinado), en tanto que ya no permitirían identificar una tupla concreta en una relación. Esta propiedad de las relaciones y de sus llaves primarias se conoce como integridad de las entidades.

2.3.5 Para la transformación entre modelos

- Toda entidad se convierte en tabla.
- Toda entidad débil se convierte en tabla. La clave de esta tabla será la mezcla de la clave del débil más la clave del fuerte.
- En una herencia, la entidad padre se convierte de forma normal. Las hijas heredan la clave del padre (y en las hijas será además clave ajena) (*Quiroz, 2003*).
- En las relaciones se trabaja de la siguiente forma:
  - Si la relación es 1:1 ambas entidades se convierten a tablas Y UNA DE ELLAS TOMA LA CLAVE DE LA OTRA que actuará solo como clave ajena.
  - Si la relación es 1:N el que tiene el N toma la clave del que tiene el 1, que actuará como parte de la clave primaria además de ser clave ajena. Si además la relación tuviera atributos, estos atributos van en el que tiene el N.
  - Si la relación es M:N LA RELACIÓN SE CONVIERTE EN TABLA. La clave de esa tabla es la mezcla de las claves de los participantes. Todas ellas actúan además como claves a. Si la relación tiene atributos los ponemos en esta tabla.

SQL

2.4.1 Como tecnología

El lenguaje SQL y los sistemas de bases de datos relacionales basados en él son una de las tecnologías de base más importantes en la industria informática actual. Durante la última década, la popularidad de SQL ha explotado y hoy en día se mantiene como el lenguaje de base de datos informático estándar. Literalmente, cientos de productos de bases de datos ahora admiten SQL y se ejecutan en sistemas informáticos, desde mainframes hasta computadoras personales e incluso dispositivos portátiles.

2.4.2 Alcance del estándar

Se ha adoptado y ampliado dos veces un estándar SQL internacional oficial. Prácticamente todos los principales productos de software empresarial dependen de SQL para la gestión de datos, y SQL es el núcleo de los productos de bases de datos de Microsoft y Oracle, dos de las empresas de software más grandes del mundo. Desde sus oscuros comienzos como un proyecto de investigación de IBM, SQL ha saltado a la fama como una tecnología informática importante y una poderosa fuerza de mercado.

2.4.3 Como herramienta

SQL es una herramienta para organizar, administrar y recuperar datos almacenados en una base de datos informática. El nombre "SQL" es una abreviatura de Structured Query Language (lenguaje de consulta estructurado). Por razones históricas, SQL suele pronunciarse "sequel", pero también se utiliza la pronunciación alternativa "S.Q.L." Como su nombre lo indica, SQL es un lenguaje informático que se utiliza para interactuar con una base de datos. De hecho, SQL funciona con un tipo específico de base de datos, llamada base de datos relacional.

2.4.4 Implementación

Cuando necesita recuperar datos de una base de datos, utiliza el lenguaje SQL para realizar la solicitud. El DBMS procesa la solicitud SQL, recupera los datos solicitados y se los devuelve. Este proceso de solicitar datos de una base de datos y recibir los resultados se denomina consulta de base de datos, de ahí el nombre de lenguaje de consulta estructurado.

2.4.5 Funciones

El nombre de lenguaje de consulta estructurado es en realidad un nombre poco apropiado. En primer lugar, SQL es mucho más que una herramienta de consulta, aunque ese era su propósito original y la recuperación de datos sigue siendo una de sus funciones más importantes. SQL se utiliza para controlar todas las funciones que un DBMS proporciona a sus usuarios, incluidas: - **Definición de datos**. SQL permite a un usuario definir la estructura y la organización de los datos almacenados y las relaciones entre los elementos de datos almacenados. - **Recuperación de datos**. SQL permite a un usuario o un programa de aplicación recuperar datos almacenados de la base de datos y utilizarlos. - **Manipulación de datos**. SQL permite que un usuario o un programa de aplicación actualice la base de datos agregando nuevos datos, eliminando datos antiguos y modificando datos almacenados previamente. - **Control de acceso**. SQL se puede utilizar para restringir la capacidad de un usuario de recuperar, agregar y modificar datos, protegiendo los datos almacenados contra el acceso no autorizado. - **Uso compartido de datos**. SQL se utiliza para coordinar el uso compartido de datos por parte de usuarios simultáneos, lo que garantiza que no interfieran entre sí. - **Integridad de los datos**. SQL define restricciones de integridad en la base de datos, protegiéndola de la corrupción debido a actualizaciones inconsistentes o fallas del sistema.

3. Herramientas empleadas

1. ERD Plus. Herramienta web para describir y graficar modelos Entidad-Relación, así como modelos relacionales, de una manera intuitiva.

2. MariaDB. MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL (General Public License). Es desarrollado por Michael (Monty) Widenius —fundador de MySQL—, la fundación MariaDB y la comunidad de desarrolladores de software libre. Introduce dos motores de almacenamiento nuevos, uno llamado Aria —que reemplaza a MyISAM— y otro llamado XtraDB —en sustitución de InnoDB—. Tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, API y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente.

4. Desarrollo

Análisis de requisitos

4.1.1 Fuentes principales de requisitos

- Interesados
- Sistemas existentes
- Documentos existentes
- Competidores y otros sistemas similares
- Interfaces con los sistemas
- Leyes y normas

4.1.2 Partes interesadas, stakeholders

1. **Cliente** – personas que pagan por el desarrollo del sistema. Son las personas que tienen la última palabra sobre cuál será el producto. Para un producto interno, ellos son los que se destacan como gerentes de producto. Además, para el mercado de consumo, el consumidor puede actuar como departamento de marketing.
2. **Usuarios** – el usuario de los productos/sistemas actuales y futuros también son partes interesadas importantes para una organización. Son los verdaderos expertos de los sistemas actuales y de la competencia. Son los mejores indicadores de mejoras en los sistemas existentes. Sus necesidades son lo que la organización debe poner en alta prioridad y no debe descuidar sus ideas y sugerencias. También debemos seleccionar cuidadosamente a nuestros usuarios.
3. **Expertos en dominios** – Son los expertos que saben de qué trabajo se trata. Ellos son los que deben estar familiarizados con los problemas que el software o sistema debe resolver. Además, conocen el entorno en el que se utilizará el producto.
4. **Inspectores** – Son los expertos en normas y reglamentos gubernamentales y en la seguridad que requiere el proyecto.
5. **Expertos en sistemas** – los expertos en sistemas, son los que interactúan con el sistema para construirlo. Están muy familiarizados con las interfaces del sistema.

4.1.3 Alcance del proyecto

Se plantea entregar en la consumación del proyecto un diseño adecuado, según las normas del modelo relacional, de un sistema de bases de datos que permita alcanzar los objetivos previstos por la organización presente; esto adecuando la información que se obtuvo en el transcurso del proyecto para generar un activo a la organización.

Modelo Entidad - Relación

En la Tabla 1, así como en la Figura 1, se presenta la propuesta de Modelo Entidad - Relación para el caso actual, el cual bajo las circunstancias del modelo de negocios presente, se modeló para cubrir las necesidades más prioritarias de la organización.

Modelo relacional

En la Figura 2 se presenta la propuesta de Modelo Entidad - Relación para el caso actual, el cual bajo las circunstancias del modelo de negocios presente, se modeló para cubrir las necesidades más prioritarias de la organización.

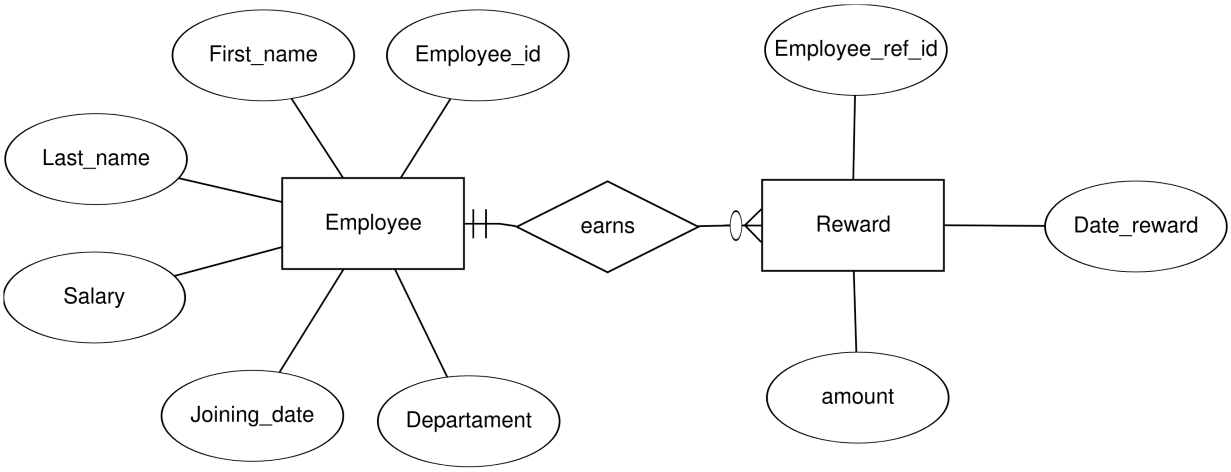

Figure 1: Modelo Entidad - Relación propuesto.

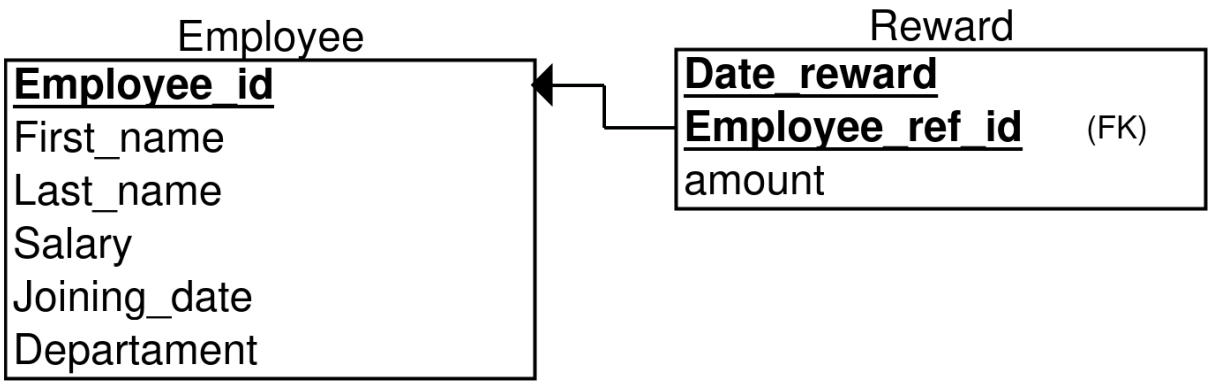

Figure 2: Modelo Relacional propuesto.

Table 1: Matriz de relaciones.

Entidades	Employee	Reward
Employee	X	X
Reward	X	X

Sentencias SQL

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

```
CREATE TABLE Employee (
        Employee_id INT NOT NULL AUTO_INCREMENT,
        First_name VARCHAR(30) NOT NULL,
        Last_name VARCHAR(30) NOT NULL,
        Salary INT NOT NULL,
        Joining_date DATE NOT NULL,
        Departament VARCHAR(20) NOT NULL,
       PRIMARY KEY (Employee_id)
);
```

```
MariaDB [EmployeeReward]> CREATE TABLE Employee (
    -> Employee_id INT NOT NULL AUTO_INCREMENT,
    -> First_name VARCHAR(30) NOT NULL,
    -> Last_name VARCHAR(30) NOT NULL,
    -> Salary INT NOT NULL,
    -> Joining_date DATE NOT NULL,
    -> Departament VARCHAR(20) NOT NULL,
    -> PRIMARY KEY (Employee_id)
    -> );
Query OK, 0 rows affected (0.006 sec)

MariaDB [EmployeeReward]>
```

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

```
INSERT INTO Employee 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", "Banking");
```

```
MariaDB [EmployeeReward]> INSERT INTO Employee 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"),
Query OK, 7 rows affected (0.002 sec)"2019-06-20", "Banking");
Records: 7  Duplicates: 0  Warnings: 0

MariaDB [EmployeeReward]>
```

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

```
CREATE TABLE Reward (
        Employee_ref_id INT NOT NULL,
        Date_reward DATE NOT NULL,
        amount INT NOT NULL,
        CONSTRAINT PK_Reward PRIMARY KEY (Employee_ref_id, Date_reward),
        FOREIGN KEY (Employee_ref_id) REFERENCES Employee(Employee_id)
);
```

```
MariaDB [EmployeeReward]> CREATE TABLE Reward (
    -> Employee_ref_id INT NOT NULL,
    -> Date_reward DATE NOT NULL,
    -> amount INT NOT NULL,
    -> CONSTRAINT PK_Reward PRIMARY KEY (Employee_ref_id, Date_reward),
    -> FOREIGN KEY (Employee_ref_id) REFERENCES Employee(Employee_id)
    -> );
Query OK, 0 rows affected, 1 warning (0.005 sec)

MariaDB [EmployeeReward]>
```

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

```
INSERT INTO Reward VALUES
(1, "2019-05-11", 1000),
(2, "2019-02-15", 5000),
(3, "2019-04-22", 2000),
(1, "2019-06-20", 8000);
```

```
MariaDB [EmployeeReward]> INSERT INTO Reward VALUES
    -> (1, "2019-05-11", 1000),
    -> (2, "2019-02-15", 5000),
    -> (3, "2019-04-22", 2000),
    -> (1, "2019-06-20", 8000);
Query OK, 4 rows affected (0.002 sec)
Records: 4  Duplicates: 0  Warnings: 0

MariaDB [EmployeeReward]>
```

Listing 5: Obtener todos los empleados.

```
SELECT *
FROM Employee;
```

```
MariaDB [EmployeeReward]> SELECT *
    -> FROM Employee;
+-------------+------------+-----------+---------+--------------+--------------+
| 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 | 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   | Banking      |
+-------------+------------+-----------+---------+--------------+--------------+
7 rows in set (0.000 sec)

MariaDB [EmployeeReward]>
```

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

```
SELECT First_name, Last_name
FROM Employee;
```

```
MariaDB [EmployeeReward]> SELECT First_name, Last_name
    -> FROM Employee;
+------------+-----------+
| First_name | Last_name |
+------------+-----------+
| Bob        | Kinto     |
| Jerry      | Kansxo    |
| Philip     | Jose      |
| John       | Abraham   |
| Michael    | Mathew    |
| Alex       | chreketo  |
| Yohan      | Soso      |
+------------+-----------+
7 rows in set (0.000 sec)

MariaDB [EmployeeReward]>
```

Listing 7: Obtener todos los valores de la columna “First_name” usando el alias “Nombre de empleado”.

```
SELECT First_name AS "Nombre de empleado"
FROM Employee;
```

```
MariaDB [EmployeeReward]> SELECT First_name AS "Nombre de empleado"
    -> FROM Employee;
+--------------------+
| Nombre de empleado |
+--------------------+
| Bob                |
| Jerry              |
| Philip             |
| John               |
| Michael            |
| Alex               |
| Yohan              |
+--------------------+
7 rows in set (0.000 sec)

MariaDB [EmployeeReward]>
```

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

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

```
MariaDB [EmployeeReward]> SELECT LOWER(Last_name)
    -> FROM Employee;
+-------------------+
| LOWER(Last_name) |
+-------------------+
| kinto             |
| kansxo            |
| jose              |
| abraham           |
| mathew            |
| chreketo          |
| soso              |
+-------------------+
7 rows in set (0.000 sec)

MariaDB [EmployeeReward]>
```

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

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

```
MariaDB [EmployeeReward]> SELECT UPPER(Last_name)
    -> FROM Employee;
+-------------------+
| UPPER(Last_name) |
+-------------------+
| KINTO             |
| KANSXO            |
| JOSE              |
| ABRAHAM           |
| MATHEW            |
| CHREKETO          |
| SOSO              |
+-------------------+
7 rows in set (0.000 sec)

MariaDB [EmployeeReward]>
```

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

```
SELECT DISTINCT Departament
FROM Employee;
```

```
MariaDB [EmployeeReward]> SELECT DISTINCT Departament
    -> FROM Employee;
+-------------+
| Departament |
+-------------+
| Finance     |
| IT          |
| Banking     |
| Insurance   |
+-------------+
4 rows in set (0.001 sec)

MariaDB [EmployeeReward]>
```

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

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

```
MariaDB [EmployeeReward]> SELECT SUBSTRING(First_name, 1, 4)
    -> FROM Employee;
+-------------------------------+
| SUBSTRING(First_name, 1, 4) |
+-------------------------------+
| Bob                           |
| Jerr                          |
| Phil                          |
| John                          |
| Mich                          |
| Alex                          |
| Yoha                          |
+-------------------------------+
7 rows in set (0.000 sec)

MariaDB [EmployeeReward]>
```

Listing 12: Obtener la posición de la letra “h” en el nombre del empleado con First_name

```
SELECT First_name, REGEXP_INSTR(First_name, "h") "Posicion de h"
FROM Employee
WHERE First_name = "Jhon";
```

```
MariaDB [EmployeeReward]> SELECT First_name, REGEXP_INSTR(First_name, "h") "Posici
on de h"
    -> FROM Employee
    -> WHERE First_name = "Jhon";
Empty set (0.000 sec)

MariaDB [EmployeeReward]>
```

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

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

```
MariaDB [EmployeeReward]> SELECT RTRIM(First_name)
    -> FROM Employee;
+--------------------+
| RTRIM(First_name) |
+--------------------+
| Bob                |
| Jerry              |
| Philip             |
| John               |
| Michael            |
| Alex               |
| Yohan              |
+--------------------+
7 rows in set (0.000 sec)

MariaDB [EmployeeReward]>
```

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

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

```
MariaDB [EmployeeReward]> SELECT LTRIM(First_name)
    -> FROM Employee;
+--------------------+
| LTRIM(First_name) |
+--------------------+
| Bob                |
| Jerry              |
| Philip             |
| John               |
| Michael            |
| Alex               |
| Yohan              |
+--------------------+
7 rows in set (0.000 sec)

MariaDB [EmployeeReward]>
```

5. Conclusiones

Con la presente práctica se demuestra día a día el alcance que los estudiantes de Bases de Datos Distribuidas pueden obtener al hacer uso de sus habilidades y conocimientos.
Algo a destacar es la versatilidad con la que las herramientas de software nos ofrece al irlas conociendo y utilizando concienzudamente, bajo entornos de desarrollo ágil, para problemáticas tanto simples como complejas, y bajo políticas de ortogonalidad y modularidad que tanto vemos día a día.
Es importante que sigamos aprendiendo a resolver problemas de las maneras más eficaces posibles.

Referencias Bibliográficas

References

Gomez, M. D. (2013). “Bases de datos”. MB Manuela Mejia, Entrevistador.

Quiroz, J. (2003). “El modelo relacional de bases de datos”. Boletín de Política Informática, 6, 53–61.