Universidad Autónoma del Estado de Hidalgo

ICBI

REPORTE DE PRÁCTICA NO. 3.1

Consultas a la BD de Distribuidora

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

El análisis de requerimientos es un proceso fundamental en el desarrollo de software que implica la identificación, documentación y gestión de las necesidades de los usuarios. Este proceso permite garantizar que el sistema final cumpla con las expectativas y requisitos establecidos. Los requerimientos pueden clasificarse en funcionales y no funcionales, y su correcta definición es crucial para el éxito del proyecto.
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.

Diseño de Vistas

El diseño de vistas se refiere a la creación de interfaces visuales que facilitan la interacción del usuario con el sistema. Esto incluye la organización de elementos gráficos, la navegación y la presentación de información. Un diseño efectivo mejora la usabilidad y la experiencia del usuario, lo que a su vez puede influir en la aceptación del software.

Diseño Conceptual

El diseño conceptual es una etapa del desarrollo que implica la creación de un modelo abstracto del sistema. Este modelo ayuda a visualizar la estructura y las relaciones entre las diferentes entidades.

2.3.1 Análisis de Entidades

El análisis de entidades consiste en identificar y definir las entidades clave que forman parte del sistema. Las entidades pueden ser objetos, personas o conceptos que tienen relevancia en el contexto del sistema. Este análisis es esencial para desarrollar un modelo de datos sólido.

2.3.2 Análisis Funcional

El análisis funcional se centra en describir las funciones y procesos que debe realizar el sistema. Esto incluye la identificación de entradas, salidas y procesos intermedios, permitiendo entender cómo el sistema debe comportarse en diferentes situaciones.

Integración de Vistas

La integración de vistas implica combinar diferentes interfaces y componentes del sistema para ofrecer una experiencia de usuario coherente. Esto requiere considerar la interactividad entre las distintas vistas y asegurar que la información se presente de manera consistente.

Esquema Conceptual Global

El esquema conceptual global es una representación visual que resume todos los elementos y relaciones del sistema en su conjunto. Este esquema sirve como guía para los desarrolladores y stakeholders, proporcionando una comprensión clara de cómo se interconectan los componentes del sistema y cómo se espera que funcione.

Modelo Entidad - Relación

2.6.1 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*).

2.6.2 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.

2.6.3 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

2.7.1 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.

2.7.2 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.

2.7.3 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).

2.7.4 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.

2.7.5 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.

Tipos y grados de relaciones

Podemos hacer dos clasificaciones diferentes de las relaciones. Una atendiendo al grado de las relaciones y otra atendiendo a la cardinalidad de la relación.

2.8.1 Grados de relaciones

Se define el **grado de la relación** cómo el número de entidades que participan en la relación (*Zambrano Ramírez, 2008*). Distinguimos tres tipos de relaciones en función del grado:

- **Reflexivas**: son relaciones de una entidad consigo misma. Por ejemplo, si tenemos una entidad alumnos en la que almacenamos todos los alumnos de un curso, incluido su delegado, aparece la relación "delegado" de la entidad alumnos consigo misma. Igual sucede en la entidad empleados si almacenamos los datos de todos los empleados de la empresa, incluidos los jefes. En este caso aparecería la relación "jefe" de la entidad empleados consigo misma.
- **Binarias**: son relaciones entre dos entidades. Por ejemplo, la relación que aparece entre clientes y pedidos o la relación que aparece entre alumnos y profesores. Son las relaciones que nos encontramos con más frecuencia.
- **Terciarias**: son relaciones entre tres entidades. Este tipo de relaciones aparece, por ejemplo, entre las entidades cuenta, sucursal y cliente para la base de datos de un banco. Cada cliente va a disponer de una o varias cuentas en una sucursal determinada, por tanto la relación afecta a las tres entidades y no sólo a dos de ellas.

2.8.2 Cardinalidad y tipos de relaciones

Se define la **cardinalidad** de la relación cómo el número de instancias de una entidad que pueden asociarse a otra entidad diferente, mediante una relación.
Distinguimos tres tipos de relaciones en función de la cardinalidad:

- **Uno a uno (1:1)**: En este tipo de relaciones, una instancia de una entidad se relaciona con una, y sólo una, instancia de otra entidad. Por ejemplo, si tenemos una base de datos para gestionar un conjunto de empresas y tenemos una entidad en la que almacenamos las empresas y otra en la que almacenamos los directores, la cardinalidad de la relación entre empresas y directores va a ser 1:1 ya que, cada empresa va a tener uno, y sólo un director y, a su vez, cada director va a dirigir una, y sólo una empresa.

- **Uno a muchos (1:M)**: En este tipo de relaciones, una instancia de una entidad se relaciona con muchas instancias de otra entidad. Sería el caso de la relación entre clientes y pedidos ya que un cliente va a realizar muchos pedidos a la empresa pero un pedido sólo puede haber sido realizado por uno, y sólo un cliente.

- **Muchos a muchos (M:M)**: En este tipo de relaciones, muchas instancias de una entidad se relacionan con muchas instancias de otra entidad. Es el caso de la relación entre alumnos y profesores. Un alumno tiene muchos profesores que le imparten clase y, a su vez, un profesor, tiene muchos alumnos a los que impartir clase.

La cardinalidad se representa en el modelo entidad relación cómo unos paréntesis que situamos justo al lado de la relación, de la siguiente forma: (1,1), (1, M) o (M:M).

Tipos de claves

Entre los componentes de las entidades, podemos encontrar atributos que identifican de manera única a la entidad, o que pueden servir para tal fin. A éstos se les conoce como **clave primaria**, **superclave** y **clave candidata** respectivamente (*Silberschatz et al., 2002*).

- **Superclave**: Conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única una entidad en el conjunto de entidades.

  Por ejemplo, el atributo id-cliente del conjunto de entidades cliente es suficiente para distinguir una entidad cliente de las otras. Así, id-cliente es una superclave. Análogamente, la combinación de nombre-cliente e id-cliente es una superclave del conjunto de entidades cliente. El atributo nombre-cliente de cliente no es una superclave, porque varias personas podrían tener el mismo nombre.

  El concepto de una superclave no es suficiente para lo que aquí se propone, ya que, como se ha visto, una superclave puede contener atributos innecesarios. Si K es una superclave, entonces también lo es cualquier superconjunto de K. A menudo interesan las superclaves tales que los subconjuntos propios de ellas no son superclave. Tales superclaves mínimas son las claves candidatas.

- **Clave candidata**: Tal y cómo hemos visto, en una entidad puede existir más de un atributo o más de un conjunto de atributos que podrían ser elegidos cómo clave primaria.

  Es posible que conjuntos distintos de atributos pudieran servir como clave candidata. Supóngase que una combinación de nombre-cliente y calle-cliente es suficiente para distinguir entre los miembros del conjunto de entidades cliente. Entonces, los conjuntos id-cliente y nombre-cliente, calle-cliente son claves candidatas. Aunque los atributos id-cliente y nombre-cliente juntos puedan distinguir entidades cliente, su combinación no forma una clave candidata, ya que el atributo id-cliente por sí solo es una clave candidata.

  Cada atributo o cada conjunto de atributos que podría ser elegido cómo clave primaria recibe el nombre de clave candidata. En el ejemplo anterior, tanto el número de expediente cómo el conjunto formado por los atributos nombre y apellidos serían claves candidatas. Del conjunto de claves candidatas se

elige una de ellas que va a ser clave primaria, en este caso el número de expediente del alumno. El resto siguen siendo claves candidatas en nuestro modelo entidad relación.

- **Clave primaria**: es el atributo o conjunto de atributos que identifican de forma única a una entidad.

  Se usará el término clave primaria para denotar una clave candidata que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades. Una clave (primaria, candidata y superclave) es una propiedad del conjunto de entidades, más que de las entidades individuales. Cualesquiera dos entidades individuales en el conjunto no pueden tener el mismo valor en sus atributos clave al mismo tiempo. La designación de una clave representa una restricción en el desarrollo del mundo real que se modela.

  La clave primaria se debería elegir de manera que sus atributos nunca, o muy raramente, cambien. Por ejemplo, el campo dirección de una persona no debería formar parte de una clave primaria, porque probablemente cambiará. Los números de D.N.I., por otra parte, es seguro que no cambiarán. Los identificadores únicos generados por empresas generalmente no cambian, excepto si se fusionan dos empresas; en tal caso el mismo identificador puede haber sido emitido por ambas empresas y es necesario la reasignación de identificadores para asegurarse de que sean únicos.

  Por ejemplo, en la entidad alumnos, la clave primaria podría ser el número de expediente del alumno.

  También podriamos tener una clave multiatributo en dicha entidad formada por el nombre y los apellidos de nuestros alumnos. Ésta elección de clave primaria sería peor que la anterior por dos motivos. Primero porque es posible, aunque no frecuente, que existan dos personas con el mismo nombre y los dos apellidos. Y, sobretodo, porque es conveniente elegir claves numéricas. Sólo existe una forma de escribir el número de expediente del alumno. Sin embargo, podemos escribir de muchas formas el nombre y los apellidos de los alumnos, podemos utilizar mayúsculas y minúsculas, podemos dejar más o menos espacios en blanco o incluso podemos escribir hasta abreviaturas. Esta arbitrariedad al escribir los atributos textuales, los hace peores candidatos a la hora de ser elegidos cómo clave primaria. Normalmente, elegimos cómo clave primaria atributos numéricos, el DNI, el número de expediente, un código, etc.

  Para distinguir, en el modelo entidad relación, la clave primaria del resto de atributos, subrayamos su nombre en la representación gráfica.

- **Clave foránea**: Referencia a clave primaria de una entidad en relación con otra.

  De manera predeterminada, una clave externa referencia los atributos que forman la clave primaria de la tabla referenciada. SQL también soporta una versión de la cláusula references, donde se puede especificar explícítamente una lista de atributos de la relación referenciada. Esta lista se debe declarar como clave candidata de la relación referenciada.

  Las claves primarias pueden especificarse como parte de la instrucción CREATE TABLE de SQL usando la cláusula FOREIGN KEY.

Fragmentación de los datos

Si la relación r se fragmenta, r se divide en varios fragmentos $r_1, r_2, ..., r_n$. Estos fragmentos contienen suficiente información como para permitir la reconstrucción de la relación original r. Hay dos esquemas diferentes de fragmentación de las relaciones: **fragmentación horizontal** y **fragmentación vertical**.
La fragmentación horizontal divide la relación asignando cada tupla de r en uno o más fragmentos. La fragmentación vertical divide la relación descomponiendo el esquema R de la relación rExisten dos esquemas diferentes:

- **Horizontal**
  - Divide la relación r asignando cada tupla de r a uno o más segmentos, cargando subconjuntos de **tuplas**.
  - Puede definirse como una selección de la relación global r, utilizando un predicado P_i para construir el fragmento r_i:

$r_i = \sigma P_i(r)$

Por ejemplo:

$r_i = \sigma_{id\_atributo=valor}(tabla)$

- La relación r se reconstruye con la unión de todos los fragmentos: $r = r_1 \cup r_2 \cup ... \cup r_n$.

- **Vertical**
  - Divide la relación descomponiendo el esquema R de la relación r, cargando subconjuntos de **atributos**.
  - En su forma más sencilla, la fragmentación vertical es igual que la descomposición.
  - Implica la definición de varios subconjuntos de atributo $R_1, R_2, ..., R_n$ del esquema R de modo que: $> R = R_1 \cup R_2 \cup ... \cup R_n$
  - Cada fragmento r_i de r se define mediante: $> r_i = \Pi_{Ri}(r)$
  - Para reconstruir la relación r se utiliza la reunión natural: $> r = r_1 \bowtie r_2 \bowtie r_3 \bowtie ... \bowtie r_n$

.

SQL

2.11.1 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.

2.11.2 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.

2.11.3 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.

2.11.4 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.

2.11.5 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.

2.11.6 ¿Qué es una inyección SQL?

La inyección SQL es el alojamiento del código malicioso en aplicaciones alojadas en páginas web con el fin de atacar esas páginas web y/o recoger los datos de los usuarios.
Además de las filtraciones de datos, pueden usar esta técnica para alimentar con información falsa la base de datos de la aplicación, retirar información importante o denegar el acceso a los propietarios o creadores de la base de datos de la aplicación.
Para hacer esto, deben encontrar y explotar alguna vulnerabilidad de la seguridad en el software de la aplicación objetivo (*Castillo, 2020*).

2.11.7 INDEX

Los índices se utilizan para encontrar rápidamente filas con valores de columna específicos.
Sin un índice, MySQL debe comenzar con la primera fila y luego leer toda la tabla para encontrar las filas relevantes. Cuanto más grande sea la tabla, más cuesta esto. Si la tabla tiene un índice para las columnas en cuestión, MySQL puede determinar rápidamente la posición que debe buscar en el medio del archivo de datos sin tener que mirar todos los datos. Esto es mucho más rápido que leer cada fila secuencialmente.

2.11.8 VIEW

Las vistas en MySQL son “tablas virtuales” que se utilizan para ver datos de una o más tablas. Las vistas no tienen sus datos, sino que almacenan datos de forma virtual, que consisten en filas y columnas.
Son muy útiles para restringir el acceso a los datos críticos de su aplicación a usuarios de terceros. Pueden ser creadas seleccionando algunas o todas las columnas y algunas o todas las filas de una tabla filtrando las filas en función de alguna(s) condición(es).
Las vistas ayudan particularmente de las siguientes maneras:

- **Simplicidad**: en lugar de escribir consultas y uniones complejas, las vistas proporcionan una forma de escribir declaraciones `SELECT` simples.
- **Seguridad mejorada**: las vistas exponen solo los datos a las aplicaciones de terceros y ocultan los detalles internos como la estructura de la tabla, los atributos, etc., lo que agrega seguridad adicional.
- **Consistencia**: al escribir vistas en lugar de consultas comunes, podemos escribir una vista que evite múltiples declaraciones y definiciones de las mismas consultas y, finalmente, proporcione una forma centralizada.

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.

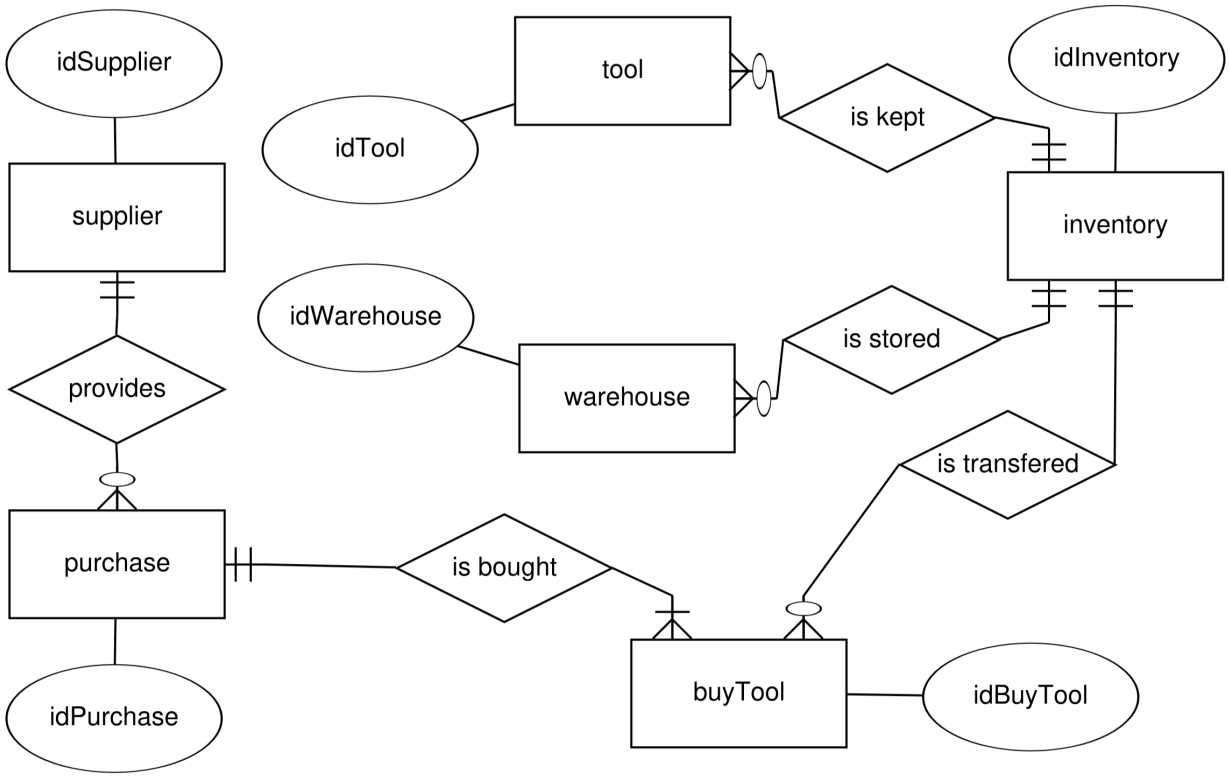


Figure 1: Modelo Entidad - Relación propuesto.

Table 1: Matriz de relaciones.

	supplier	tool	purchase	buyTool	warehouse	inventory
supplier			X			
tool						X
purchase	X			X		
buyTool			X			X
warehouse						X
inventory		X		X	X	

Sentencias SQL

Listing 1: Creación de base de datos.

```
CREATE DATABASE distribuidora;
USE distribuidora;
```

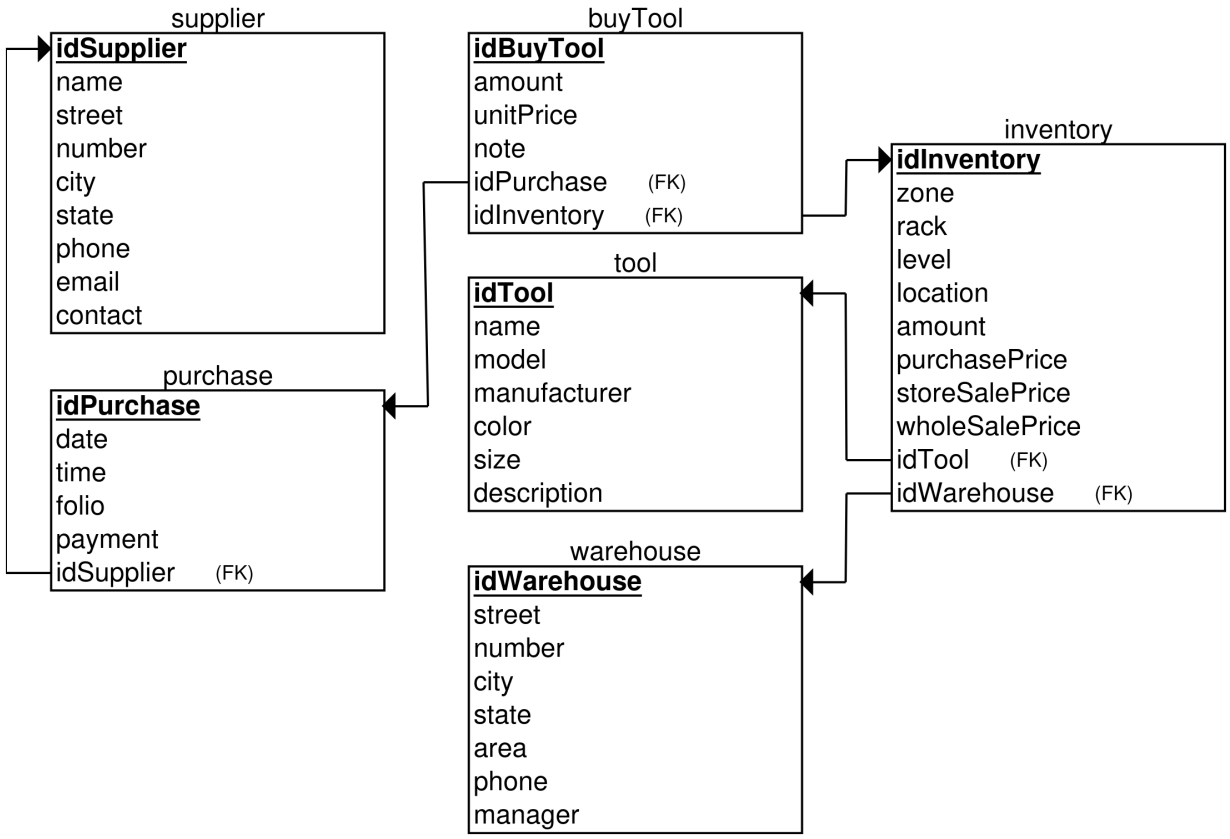


Figure 2: Modelo Relacional propuesto.

```
MariaDB [(none)]> CREATE DATABASE distribuidora;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> USE distribuidora;
Database changed
MariaDB [distribuidora]>
```

Listing 2: Creación de tabla supplier.

```
CREATE TABLE supplier (
 idSupplier INT NOT NULL AUTO_INCREMENT,
 name VARCHAR(100) NOT NULL,
 street VARCHAR(100) NOT NULL,
 number VARCHAR(10) NOT NULL,
 city VARCHAR(50) NOT NULL,
 state VARCHAR(50) NOT NULL,
 phone VARCHAR(20) NOT NULL,
 email VARCHAR(100) NOT NULL,
 contact VARCHAR(100) NOT NULL,
PRIMARY KEY (idSupplier)
```

```
);
```

```
MariaDB [distribuidora]> CREATE TABLE supplier (
    ->     idSupplier INT NOT NULL AUTO_INCREMENT,
    ->     name VARCHAR(100) NOT NULL,
    ->     street VARCHAR(100) NOT NULL,
    ->     number VARCHAR(10) NOT NULL,
    ->     city VARCHAR(50) NOT NULL,
    ->     state VARCHAR(50) NOT NULL,
    ->     phone VARCHAR(20) NOT NULL,
    ->     email VARCHAR(100) NOT NULL,
    ->     contact VARCHAR(100) NOT NULL,
    ->     PRIMARY KEY (idSupplier)
    -> );
Query OK, 0 rows affected (0.004 sec)
```

Listing 3: Población de tabla supplier.

```
INSERT INTO supplier (name, street, number, city, state, phone, email, contact) VA
("Herramientas del Norte", "Calle 15", "123", "Monterrey", "Nuevo León", "818-123-
("Proveedores de Herramientas S.A.", "Av. Revolución", "456", "Ciudad de México",
("Distribuciones de Herramientas", "Calle 5 de Febrero", "789", "Guadalajara", "Ja
("Soluciones en Herramientas", "Calle 8", "321", "Puebla", "Puebla", "222-456-7890
("Importaciones y Herramientas", "Av. 20 de Noviembre", "654", "Tijuana", "Baja Ca
```

```
MariaDB [distribuidora]> INSERT INTO supplier (name, street, number, city, state, phone, email, contact) VALUES

    -> ("Herramientas del Norte", "Calle 15", "123", "Monterrey", "Nuevo León", "818-123-4567", "ventas@herrami
entasnorte.com", "Alberto Ruiz"),
    -> ("Proveedores de Herramientas S.A.", "Av. Revolución", "456", "Ciudad de México", "CDMX", "555-234-5678"
, "contactoprovherr@hotmail.com", "Sofía Gómez"),
    -> ("Distribuciones de Herramientas", "Calle 5 de Febrero", "789", "Guadalajara", "Jalisco", "333-345-6789"
, "infodistribucionesherramientas@gmail.com", "Carlos Mendoza"),
    -> ("Soluciones en Herramientas", "Calle 8", "321", "Puebla", "Puebla", "222-456-7890", "soporte@soluciones
herramientas.com", "María Elena Pérez"),
    -> ("Importaciones y Herramientas", "Av. 20 de Noviembre", "654", "Tijuana", "Baja California", "664-567-89
01", "ventas@toolimport.com", "Luis Miguel Torres");
Query OK, 5 rows affected (0.002 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Listing 4: Creación de tabla purchase.

```
CREATE TABLE purchase (
idPurchase INT NOT NULL AUTO_INCREMENT,
idSupplier INT NOT NULL,
date DATE NOT NULL,
time TIME NOT NULL,
folio VARCHAR(50) NOT NULL,
payment DECIMAL(10, 2) NOT NULL,
PRIMARY KEY (idPurchase),
FOREIGN KEY (idSupplier) REFERENCES supplier(idSupplier)
);
```

```
MariaDB [distribuidora]> CREATE TABLE purchase (
    ->     idPurchase INT NOT NULL AUTO_INCREMENT,
    ->     idSupplier INT NOT NULL,
    ->     date DATE NOT NULL,
    ->     time TIME NOT NULL,
    ->     folio VARCHAR(50) NOT NULL,
    ->     payment DECIMAL(10, 2) NOT NULL,
    ->     PRIMARY KEY (idPurchase),
    ->     FOREIGN KEY (idSupplier) REFERENCES supplier(idSupplier)
    -> );
Query OK, 0 rows affected (0.006 sec)
```

Listing 5: Población de tabla purchase.

```
INSERT INTO purchase (idSupplier, date, time, folio, payment) VALUES
(1, "2024-01-15", "10:30:00", "FOL-2983", 1520.00),
(2, "2024-01-18", "14:15:00", "FOL-1756", 2500.00),
(3, "2024-01-20", "09:00:00", "FOL-4821", 1250.00),
(4, "2024-02-02", "11:45:00", "FOL-6378", 800.00),
(5, "2024-02-05", "13:30:00", "FOL-9034", 3000.00),
(1, "2024-02-10", "16:00:00", "FOL-2045", 1860.00),
(2, "2024-02-12", "08:20:00", "FOL-8746", 2200.00),
(3, "2024-02-15", "12:00:00", "FOL-3950", 950.00),
(4, "2024-02-18", "10:00:00", "FOL-5123", 500.00),
(5, "2024-03-01", "15:00:00", "FOL-7489", 4510.00);
```

```
MariaDB [distribuidora]> INSERT INTO purchase (idSupplier, date, time, folio, payment) VALUES
    -> (1, "2024-01-15", "10:30:00", "FOL-2983", 1520.00),
    -> (2, "2024-01-18", "14:15:00", "FOL-1756", 2500.00),
    -> (3, "2024-01-20", "09:00:00", "FOL-4821", 1250.00),
    -> (4, "2024-02-02", "11:45:00", "FOL-6378", 800.00),
    -> (5, "2024-02-05", "13:30:00", "FOL-9034", 3000.00),
    -> (1, "2024-02-10", "16:00:00", "FOL-2045", 1860.00),
    -> (2, "2024-02-12", "08:20:00", "FOL-8746", 2200.00),
    -> (3, "2024-02-15", "12:00:00", "FOL-3950", 950.00),
    -> (4, "2024-02-18", "10:00:00", "FOL-5123", 500.00),
    -> (5, "2024-03-01", "15:00:00", "FOL-7489", 4510.00);
Query OK, 10 rows affected (0.002 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Listing 6: Creación de tabla tool.

```
CREATE TABLE tool (
idTool INT NOT NULL AUTO_INCREMENT,
name VARCHAR(255) NOT NULL,
model VARCHAR(100) NOT NULL,
manufacturer VARCHAR(100) NOT NULL,
color VARCHAR(50) NOT NULL,
size VARCHAR(50) NOT NULL,
description TEXT NOT NULL,
PRIMARY KEY (idTool)
);
```

```
MariaDB [distribuidora]> CREATE TABLE tool (
    ->     idTool INT NOT NULL AUTO_INCREMENT,
    ->     name VARCHAR(255) NOT NULL,
    ->     model VARCHAR(100) NOT NULL,
    ->     manufacturer VARCHAR(100) NOT NULL,
    ->     color VARCHAR(50) NOT NULL,
    ->     size VARCHAR(50) NOT NULL,
    ->     description TEXT NOT NULL,
    ->     PRIMARY KEY (idTool)
    -> );
Query OK, 0 rows affected (0.005 sec)
```

Listing 7: Población de tabla tool.

```
INSERT INTO tool (name, model, manufacturer, color, size, description) VALUES
("Taladro Inalámbrico", "DCD771C2", "DeWalt", "Amarillo", "M", "Taladro inalámbrico
("Sierra Circular", "HS7600", "Makita", "Verde", "L", "Sierra circular de 7-1/4 pu
("Juego de Destornilladores", "DR-150", "Stanley", "Negro", "S", "Juego de 15 dest
("Martillo de Framing", "51535", "Estwing", "Rojo", "M", "Martillo de 16 oz con ma
("Llave de Ajuste", "DWMT19250", "Dewalt", "Gris", "M", "Llave ajustable de 10 pulg
```

```
MariaDB [distribuidora]> INSERT INTO tool (name, model, manufacturer, color, size, description) VALUES
    -> ("Taladro Inalámbrico", "DCD771C2", "DeWalt", "Amarillo", "M", "Taladro inalámbrico compacto con batería
 de 20V, ideal para uso en casa y trabajos ligeros."),
    -> ("Sierra Circular", "HS7600", "Makita", "Verde", "L", "Sierra circular de 7-1/4 pulgadas, perfecta para
cortes en madera y materiales similares."),
    -> ("Juego de Destornilladores", "DR-150", "Stanley", "Negro", "S", "Juego de 15 destornilladores de precis
ión, ideal para reparaciones electrónicas."),
    -> ("Martillo de Framing", "51535", "Estwing", "Rojo", "M", "Martillo de 16 oz con mango de cuero, excelent
e para trabajos de construcción."),
    -> ("Llave de Ajuste", "DWMT19250", "Dewalt", "Gris", "M", "Llave ajustable de 10 pulgadas, fabricada en ac
ero al carbono para mayor durabilidad.");
Query OK, 5 rows affected (0.002 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Listing 8: Creación de tabla warehouse.

```
CREATE TABLE warehouse (
idWarehouse INT NOT NULL AUTO_INCREMENT,
street VARCHAR(255) NOT NULL,
number VARCHAR(10) NOT NULL,
city VARCHAR(100) NOT NULL,
state VARCHAR(100) NOT NULL,
area DECIMAL(10, 2) NOT NULL,
phone VARCHAR(20) NOT NULL,
manager VARCHAR(100) NOT NULL,
PRIMARY KEY (idWarehouse)
);
```

```
MariaDB [distribuidora]> CREATE TABLE warehouse (
    ->     idWarehouse INT NOT NULL AUTO_INCREMENT,
    ->     street VARCHAR(255) NOT NULL,
    ->     number VARCHAR(10) NOT NULL,
    ->     city VARCHAR(100) NOT NULL,
    ->     state VARCHAR(100) NOT NULL,
    ->     area DECIMAL(10, 2) NOT NULL,
    ->     phone VARCHAR(20) NOT NULL,
    ->     manager VARCHAR(100) NOT NULL,
    ->     PRIMARY KEY (idWarehouse)
    -> );
Query OK, 0 rows affected (0.004 sec)
```

Listing 9: Población de tabla warehouse.

```
INSERT INTO warehouse (street, number, city, state, area, phone, manager) VALUES
("Av. Juárez", "100", "Ciudad de México", "CDMX", 150.5, "555-1001-100", "Ricardo C
("Calle 15", "200", "Guadalajara", "Jalisco", 200.0, "333-1002-200", "Patricia Ram
("Calle 10", "300", "Monterrey", "Nuevo León", 250.0, "818-1003-300", "Fernando To
("Av. Insurgentes", "400", "Puebla", "Puebla", 300.5, "222-1004-400", "Isabel Cast
("Calle 5", "500", "Tijuana", "Baja California", 350.0, "664-1005-500", "Javier Ma
```

```
MariaDB [distribuidora]> CREATE TABLE inventory (
    ->     idInventory INT NOT NULL AUTO_INCREMENT,
    ->     idWarehouse INT NOT NULL,
    ->     idTool INT NOT NULL,
    ->     zone VARCHAR(100) NOT NULL,
    ->     rack VARCHAR(100) NOT NULL,
    ->     level INT NOT NULL,
    ->     location VARCHAR(100) NOT NULL,
    ->     amount INT NOT NULL,
    ->     purchasePrice DECIMAL(10, 2) NOT NULL,
    ->     storeSalePrice DECIMAL(10, 2) NOT NULL,
    ->     wholeSalePrice DECIMAL(10, 2) NOT NULL,
    ->     PRIMARY KEY (idInventory),
    ->     FOREIGN KEY (idWarehouse) REFERENCES warehouse(idWarehouse),
    ->     FOREIGN KEY (idTool) REFERENCES tool(idTool)
    -> );
Query OK, 0 rows affected (0.007 sec)
```

Listing 10: Creación de tabla inventario.

```
CREATE TABLE inventory (
idInventory INT NOT NULL AUTO_INCREMENT,
idWarehouse INT NOT NULL,
idTool INT NOT NULL,
zone VARCHAR(100) NOT NULL,
rack VARCHAR(100) NOT NULL,
```

```
level INT NOT NULL,
location VARCHAR(100) NOT NULL,
amount INT NOT NULL,
purchasePrice DECIMAL(10, 2) NOT NULL,
storeSalePrice DECIMAL(10, 2) NOT NULL,
wholeSalePrice DECIMAL(10, 2) NOT NULL,
PRIMARY KEY (idInventory),
FOREIGN KEY (idWarehouse) REFERENCES warehouse(idWarehouse),
FOREIGN KEY (idTool) REFERENCES tool(idTool)
);
```

```
MariaDB [distribuidora]> INSERT INTO inventory (idWarehouse, idTool, zone, rack, level, location, amount, purchasePrice, store
SalePrice, wholeSalePrice) VALUES
    -> (1, 1, "Zona Norte", "Rack A", 1, "Ubicación A1", 30, 600.00, 900.00, 550.00),
    -> (1, 2, "Zona Norte", "Rack A", 1, "Ubicación A2", 20, 1200.00, 1800.00, 1100.00),
    -> (1, 3, "Zona Centro", "Rack B", 1, "Ubicación B1", 15, 250.00, 400.00, 230.00),
    -> (1, 4, "Zona Centro", "Rack B", 2, "Ubicación B2", 10, 1500.00, 2200.00, 1400.00),
    -> (1, 5, "Zona Sur", "Rack C", 1, "Ubicación C1", 25, 300.00, 450.00, 280.00),
    -> (2, 1, "Zona Norte", "Rack A", 1, "Ubicación A3", 22, 600.00, 900.00, 550.00),
    -> (2, 2, "Zona Norte", "Rack A", 2, "Ubicación A4", 18, 1200.00, 1800.00, 1100.00),
    -> (2, 3, "Zona Centro", "Rack B", 1, "Ubicación B3", 28, 250.00, 400.00, 230.00),
    -> (2, 4, "Zona Centro", "Rack B", 2, "Ubicación B4", 12, 1500.00, 2200.00, 1400.00),
    -> (2, 5, "Zona Sur", "Rack C", 1, "Ubicación C2", 35, 300.00, 450.00, 280.00),
    -> (3, 1, "Zona Este", "Rack D", 1, "Ubicación D1", 10, 600.00, 900.00, 550.00),
    -> (3, 2, "Zona Este", "Rack D", 1, "Ubicación D2", 20, 1200.00, 1800.00, 1100.00),
    -> (3, 3, "Zona Centro", "Rack E", 1, "Ubicación E1", 15, 250.00, 400.00, 230.00),
    -> (3, 4, "Zona Centro", "Rack E", 2, "Ubicación E2", 8, 1500.00, 2200.00, 1400.00),
    -> (3, 5, "Zona Oeste", "Rack F", 1, "Ubicación F1", 12, 300.00, 450.00, 280.00),
    -> (4, 1, "Zona Norte", "Rack A", 1, "Ubicación A5", 25, 600.00, 900.00, 550.00),
    -> (4, 2, "Zona Sur", "Rack C", 1, "Ubicación C3", 10, 1200.00, 1800.00, 1100.00),
    -> (4, 3, "Zona Este", "Rack D", 1, "Ubicación D3", 18, 250.00, 400.00, 230.00),
    -> (4, 4, "Zona Oeste", "Rack F", 2, "Ubicación F2", 5, 1500.00, 2200.00, 1400.00),
    -> (4, 5, "Zona Centro", "Rack E", 1, "Ubicación E3", 30, 300.00, 450.00, 280.00),
    -> (5, 1, "Zona Sur", "Rack C", 1, "Ubicación C4", 35, 600.00, 900.00, 550.00),
    -> (5, 2, "Zona Este", "Rack D", 1, "Ubicación D4", 12, 1200.00, 1800.00, 1100.00),
    -> (5, 3, "Zona Oeste", "Rack F", 1, "Ubicación F3", 15, 250.00, 400.00, 230.00),
    -> (5, 4, "Zona Centro", "Rack E", 2, "Ubicación E4", 10, 1500.00, 2200.00, 1400.00),
    -> (5, 5, "Zona Norte", "Rack A", 1, "Ubicación A6", 28, 300.00, 450.00, 280.00);
Query OK, 25 rows affected (0.002 sec)
Records: 25  Duplicates: 0  Warnings: 0
```

Listing 11: Población de tabla inventario.

```
INSERT INTO inventory (idWarehouse, idTool, zone, rack, level, location, amount, p
(1, 1, "Zona Norte", "Rack A", 1, "Ubicación A1", 30, 600.00, 900.00, 550.00),
(1, 2, "Zona Norte", "Rack A", 1, "Ubicación A2", 20, 1200.00, 1800.00, 1100.00),
(1, 3, "Zona Centro", "Rack B", 1, "Ubicación B1", 15, 250.00, 400.00, 230.00),
(1, 4, "Zona Centro", "Rack B", 2, "Ubicación B2", 10, 1500.00, 2200.00, 1400.00),
(1, 5, "Zona Sur", "Rack C", 1, "Ubicación C1", 25, 300.00, 450.00, 280.00),
(2, 1, "Zona Norte", "Rack A", 1, "Ubicación A3", 22, 600.00, 900.00, 550.00),
(2, 2, "Zona Norte", "Rack A", 2, "Ubicación A4", 18, 1200.00, 1800.00, 1100.00),
(2, 3, "Zona Centro", "Rack B", 1, "Ubicación B3", 28, 250.00, 400.00, 230.00),
(2, 4, "Zona Centro", "Rack B", 2, "Ubicación B4", 12, 1500.00, 2200.00, 1400.00),
(2, 5, "Zona Sur", "Rack C", 1, "Ubicación C2", 35, 300.00, 450.00, 280.00),
(3, 1, "Zona Este", "Rack D", 1, "Ubicación D1", 10, 600.00, 900.00, 550.00),
(3, 2, "Zona Este", "Rack D", 1, "Ubicación D2", 20, 1200.00, 1800.00, 1100.00),
(3, 3, "Zona Centro", "Rack E", 1, "Ubicación E1", 15, 250.00, 400.00, 230.00),
(3, 4, "Zona Centro", "Rack E", 2, "Ubicación E2", 8, 1500.00, 2200.00, 1400.00),
(3, 5, "Zona Oeste", "Rack F", 1, "Ubicación F1", 12, 300.00, 450.00, 280.00),
(4, 1, "Zona Norte", "Rack A", 1, "Ubicación A5", 25, 600.00, 900.00, 550.00),
(4, 2, "Zona Sur", "Rack C", 1, "Ubicación C3", 10, 1200.00, 1800.00, 1100.00),
(4, 3, "Zona Este", "Rack D", 1, "Ubicación D3", 18, 250.00, 400.00, 230.00),
(4, 4, "Zona Oeste", "Rack F", 2, "Ubicación F2", 5, 1500.00, 2200.00, 1400.00),
(4, 5, "Zona Centro", "Rack E", 1, "Ubicación E3", 30, 300.00, 450.00, 280.00),
(5, 1, "Zona Sur", "Rack C", 1, "Ubicación C4", 35, 600.00, 900.00, 550.00),
```

```
(5, 2, "Zona Este", "Rack D", 1, "Ubicación D4", 12, 1200.00, 1800.00, 1100.00),
(5, 3, "Zona Oeste", "Rack F", 1, "Ubicación F3", 15, 250.00, 400.00, 230.00),
(5, 4, "Zona Centro", "Rack E", 2, "Ubicación E4", 10, 1500.00, 2200.00, 1400.00),
(5, 5, "Zona Norte", "Rack A", 1, "Ubicación A6", 28, 300.00, 450.00, 280.00);
```

```
MariaDB [distribuidora]> CREATE TABLE buyTool (
    ->     idBuyTool INT NOT NULL AUTO_INCREMENT,
    ->     idPurchase INT NOT NULL,
    ->     idInventory INT NOT NULL,
    ->     amount INT NOT NULL,
    ->     unitPrice DECIMAL(10, 2) NOT NULL,
    ->     note TEXT NOT NULL,
    ->     PRIMARY KEY (idBuyTool),
    ->     FOREIGN KEY (idPurchase) REFERENCES purchase(idPurchase),
    ->     FOREIGN KEY (idInventory) REFERENCES inventory(idInventory)
    -> );
Query OK, 0 rows affected (0.007 sec)
```

Listing 12: Creación de tabla buyTool.

```
CREATE TABLE buyTool (
idBuyTool INT NOT NULL AUTO_INCREMENT,
idPurchase INT NOT NULL,
idInventory INT NOT NULL,
amount INT NOT NULL,
unitPrice DECIMAL(10, 2) NOT NULL,
note TEXT NOT NULL,
PRIMARY KEY (idBuyTool),
FOREIGN KEY (idPurchase) REFERENCES purchase(idPurchase),
FOREIGN KEY (idInventory) REFERENCES inventory(idInventory)
);
```

```
MariaDB [distribuidora]> INSERT INTO buyTool (idBuyTool, idPurchase, idInventory, amount, unitPrice, note) VALUES
    -> (1, 1, 1, 5, 600.00, "Compra de prueba 1"),
    -> (2, 1, 2, 3, 1200.00, "Urgente para proyecto"),
    -> (3, 2, 3, 10, 250.00, "Reabastecimiento regular"),
    -> (4, 2, 4, 2, 1500.00, "Compra de herramientas especializadas"),
    -> (5, 3, 5, 7, 300.00, "Reparaciones en curso"),
    -> (6, 3, 6, 5, 600.00, "Compra mensual"),
    -> (7, 4, 7, 8, 1200.00, "Para proyecto específico"),
    -> (8, 4, 8, 4, 1800.00, "Urgente"),
    -> (9, 5, 9, 3, 450.00, "Para uso inmediato"),
    -> (10, 5, 10, 6, 300.00, "Compra para nuevo cliente"),
    -> (11, 6, 11, 2, 600.00, "Reabastecimiento"),
    -> (12, 6, 12, 10, 1200.00, "Compra especial"),
    -> (13, 7, 13, 1, 250.00, "Herramienta de alta demanda"),
    -> (14, 7, 14, 5, 1500.00, "Para mantenimiento"),
    -> (15, 8, 15, 4, 300.00, "Venta al por mayor"),
    -> (16, 8, 16, 9, 600.00, "Nuevo cliente"),
    -> (17, 9, 17, 3, 1200.00, "Reabastecimiento urgente"),
    -> (18, 9, 18, 2, 1800.00, "Herramienta crítica"),
    -> (19, 10, 19, 5, 450.00, "Para uso en campo"),
    -> (20, 10, 20, 1, 300.00, "Muestra para cliente"),
    -> (21, 1, 21, 4, 600.00, "Compra por demanda"),
    -> (22, 1, 22, 7, 1200.00, "Para proyecto de construcción"),
    -> (23, 2, 23, 8, 250.00, "Para reabastecimiento"),
    -> (24, 2, 24, 6, 1500.00, "Necesidad urgente"),
    -> (25, 3, 25, 2, 300.00, "Pedido especial"),
```

Listing 13: Población de tabla buyTool.

```
INSERT INTO buyTool (idBuyTool, idPurchase, idInventory, amount, unitPrice, note) V
(1, 1, 1, 5, 600.00, "Compra de prueba 1"),
(2, 1, 2, 3, 1200.00, "Urgente para proyecto"),
(3, 2, 3, 10, 250.00, "Reabastecimiento regular"),
(4, 2, 4, 2, 1500.00, "Compra de herramientas especializadas"),
(5, 3, 5, 7, 300.00, "Reparaciones en curso"),
(6, 3, 6, 5, 600.00, "Compra mensual"),
(7, 4, 7, 8, 1200.00, "Para proyecto específico"),
(8, 4, 8, 4, 1800.00, "Urgente"),
(9, 5, 9, 3, 450.00, "Para uso inmediato"),
(10, 5, 10, 6, 300.00, "Compra para nuevo cliente"),
(11, 6, 11, 2, 600.00, "Reabastecimiento"),
(12, 6, 12, 10, 1200.00, "Compra especial"),
(13, 7, 13, 1, 250.00, "Herramienta de alta demanda"),
(14, 7, 14, 5, 1500.00, "Para mantenimiento"),
(15, 8, 15, 4, 300.00, "Venta al por mayor"),
(16, 8, 16, 9, 600.00, "Nuevo cliente"),
(17, 9, 17, 3, 1200.00, "Reabastecimiento urgente"),
(18, 9, 18, 2, 1800.00, "Herramienta crítica"),
(19, 10, 19, 5, 450.00, "Para uso en campo"),
(20, 10, 20, 1, 300.00, "Muestra para cliente"),
(21, 1, 21, 4, 600.00, "Compra por demanda"),
(22, 1, 22, 7, 1200.00, "Para proyecto de construcción"),
(23, 2, 23, 8, 250.00, "Para reabastecimiento"),
(24, 2, 24, 6, 1500.00, "Necesidad urgente"),
(25, 3, 25, 2, 300.00, "Pedido especial"),
(26, 3, 1, 9, 600.00, "Reemplazo de herramientas viejas"),
(27, 4, 2, 3, 1200.00, "Proyecto de remodelación"),
(28, 4, 3, 4, 1800.00, "Compra para nuevo contrato"),
(29, 5, 4, 1, 450.00, "Necesidad de última hora"),
(30, 5, 5, 5, 300.00, "Para mantenimiento rutinario"),
(31, 6, 6, 6, 600.00, "Compra por demanda"),
(32, 6, 7, 3, 1200.00, "Compra mensual"),
(33, 7, 8, 5, 250.00, "Herramienta de alta rotación"),
(34, 7, 9, 4, 1500.00, "Urgente para cliente"),
(35, 8, 10, 9, 300.00, "Para prueba de calidad"),
(36, 8, 11, 7, 600.00, "Reabastecimiento"),
(37, 9, 12, 2, 1200.00, "Para reparación de maquinaria"),
(38, 9, 13, 8, 1800.00, "Requisito de proyecto"),
(39, 10, 14, 5, 450.00, "Compra mensual"),
(40, 10, 15, 3, 300.00, "Urgente para cliente"),
(41, 1, 16, 4, 600.00, "Para proyecto de construcción"),
(42, 1, 17, 2, 1200.00, "Urgente para entrega"),
(43, 2, 18, 7, 250.00, "Reabastecimiento"),
(44, 2, 19, 8, 1500.00, "Para cliente especial"),
(45, 3, 20, 6, 300.00, "Necesidad de última hora"),
(46, 3, 21, 4, 600.00, "Compra por demanda"),
(47, 4, 22, 3, 1200.00, "Proyecto de construcción"),
(48, 4, 23, 2, 1800.00, "Reabastecimiento urgente"),
(49, 5, 24, 1, 450.00, "Para uso inmediato"),
(50, 5, 25, 5, 300.00, "Reemplazo de herramientas viejas");
```

```
    → (26, 3, 1, 9, 600.00, "Reemplazo de herramientas viejas"),
    → (27, 4, 2, 3, 1200.00, "Proyecto de remodelación"),
    → (28, 4, 3, 4, 1800.00, "Compra para nuevo contrato"),
    → (29, 5, 4, 1, 450.00, "Necesidad de última hora"),
    → (30, 5, 5, 5, 300.00, "Para mantenimiento rutinario"),
    → (31, 6, 6, 6, 600.00, "Compra por demanda"),
    → (32, 6, 7, 3, 1200.00, "Compra mensual"),
    → (33, 7, 8, 5, 250.00, "Herramienta de alta rotación"),
    → (34, 7, 9, 4, 1500.00, "Urgente para cliente"),
    → (35, 8, 10, 9, 300.00, "Para prueba de calidad"),
    → (36, 8, 11, 7, 600.00, "Reabastecimiento"),
    → (37, 9, 12, 2, 1200.00, "Para reparación de maquinaria"),
    → (38, 9, 13, 8, 1800.00, "Requisito de proyecto"),
    → (39, 10, 14, 5, 450.00, "Compra mensual"),
    → (40, 10, 15, 3, 300.00, "Urgente para cliente"),
    → (41, 1, 16, 4, 600.00, "Para proyecto de construcción"),
    → (42, 1, 17, 2, 1200.00, "Urgente para entrega"),
    → (43, 2, 18, 7, 250.00, "Reabastecimiento"),
    → (44, 2, 19, 8, 1500.00, "Para cliente especial"),
    → (45, 3, 20, 6, 300.00, "Necesidad de última hora"),
    → (46, 3, 21, 4, 600.00, "Compra por demanda"),
    → (47, 4, 22, 3, 1200.00, "Proyecto de construcción"),
    → (48, 4, 23, 2, 1800.00, "Reabastecimiento urgente"),
    → (49, 5, 24, 1, 450.00, "Para uso inmediato"),
    → (50, 5, 25, 5, 300.00, "Reemplazo de herramientas viejas");
Query OK, 50 rows affected (0.003 sec)
Records: 50  Duplicates: 0  Warnings: 0
```

Listing 14: Lista de compras realizadas en el mes de enero que incluya el nombre del proveedor el nombre de la herramienta cantidad precio unitario y precio total ordenado por fecha.

```
SELECT
p.date AS Fecha,
s.name AS Proveedor,
t.name AS Herramienta,
b.amount AS Cantidad,
b.unitPrice AS "Precio unitario",
b.amount * b.unitPrice AS "Precio total"
FROM buyTool AS b
INNER JOIN purchase AS p
ON b.idPurchase = p.idPurchase
INNER JOIN supplier AS s
ON b.idSupplier = s.idSupplier
INNER JOIN tool AS t
ON b.idTool = t.idTool
HAVING MONTH(Fecha) = 01
ORDER BY Fecha;
```

$$\pi_{Mes,Nombredelproveedor,Nombredelaherramienta,Cantidad,Preciounitario,Preciototal}\left(\sigma_{\text{Mes=01}}\left(\tau_{\text{MONTH}(p.date)}\right.\right.$$
$$\left.\left.\times((p\bowtie_{p.idSupplier=s.idSupplier}s)\bowtie_{p.idPurchase=b.idPurchase}b\bowtie_{b.idInventory=i.idInventory}i\bowtie_{i.idTool=t.idTool}t)\right)\right) \tag{1}$$

```
MariaDB [distribuidora]> INSERT INTO buyTool (idBuyTool, idPurchase, idInventory, amount, unitPrice, note) VALUES
    -> (1, 1, 1, 5, 600.00, "Compra de prueba 1"),
    -> (2, 1, 2, 3, 1200.00, "Urgente para proyecto"),
    -> (3, 2, 3, 10, 250.00, "Reabastecimiento regular"),
    -> (4, 2, 4, 2, 1500.00, "Compra de herramientas especializadas"),
    -> (5, 3, 5, 7, 300.00, "Reparaciones en curso"),
    -> (6, 3, 6, 5, 600.00, "Compra mensual"),
    -> (7, 4, 7, 8, 1200.00, "Para proyecto específico"),
    -> (8, 4, 8, 4, 1800.00, "Urgente"),
    -> (9, 5, 9, 3, 450.00, "Para uso inmediato"),
    -> (10, 5, 10, 6, 300.00, "Compra para nuevo cliente"),
    -> (11, 6, 11, 2, 600.00, "Reabastecimiento"),
    -> (12, 6, 12, 10, 1200.00, "Compra especial"),
    -> (13, 7, 13, 1, 250.00, "Herramienta de alta demanda"),
    -> (14, 7, 14, 5, 1500.00, "Para mantenimiento"),
    -> (15, 8, 15, 4, 300.00, "Venta al por mayor"),
    -> (16, 8, 16, 9, 600.00, "Nuevo cliente"),
    -> (17, 9, 17, 3, 1200.00, "Reabastecimiento urgente"),
    -> (18, 9, 18, 2, 1800.00, "Herramienta crítica"),
    -> (19, 10, 19, 5, 450.00, "Para uso en campo"),
    -> (20, 10, 20, 1, 300.00, "Muestra para cliente"),
    -> (21, 1, 21, 4, 600.00, "Compra por demanda"),
    -> (22, 1, 22, 7, 1200.00, "Para proyecto de construcción"),
    -> (23, 2, 23, 8, 250.00, "Para reabastecimiento"),
    -> (24, 2, 24, 6, 1500.00, "Necesidad urgente"),
    -> (25, 3, 25, 2, 300.00, "Pedido especial"),
```

Listing 15: Listado del inventario de la bodega 1 que incluya el nombre de la herramienta cantidad y costo total (precio de compra * cantidad)

```
SELECT
w.idWarehouse AS Bodega,
t.name AS Herramienta,
b.amount AS Cantidad,
b.amount * b.unitPrice AS "Cost total"
FROM buyTool AS b
INNER JOIN inventory AS i
ON b.idBuyTool = i.idBuyTool
INNER JOIN warehouse AS w
ON i.idWarehouse = w.idWarehouse
INNER JOIN tool AS t
ON b.idTool = t.idTool
WHERE w.idWarehouse = 1;
```

$$\pi_{\text{Bodega, Herramienta, Cantidad, Cost total}}\left(\sigma_{w.idWarehouse=1}\left((b\bowtie_{b.idBuyTool=i.idBuyTool}i)\bowtie_{i.idWarehouse=w.idWarehouse}w\bowtie_{b.idTool=}\right.\right. \tag{2}$$

```
    -> (26, 3, 1, 9, 600.00, "Reemplazo de herramientas viejas"),
    -> (27, 4, 2, 3, 1200.00, "Proyecto de remodelación"),
    -> (28, 4, 3, 4, 1800.00, "Compra para nuevo contrato"),
    -> (29, 5, 4, 1, 450.00, "Necesidad de última hora"),
    -> (30, 5, 5, 5, 300.00, "Para mantenimiento rutinario"),
    -> (31, 6, 6, 6, 600.00, "Compra por demanda"),
    -> (32, 6, 7, 3, 1200.00, "Compra mensual"),
    -> (33, 7, 8, 5, 250.00, "Herramienta de alta rotación"),
    -> (34, 7, 9, 4, 1500.00, "Urgente para cliente"),
    -> (35, 8, 10, 9, 300.00, "Para prueba de calidad"),
    -> (36, 8, 11, 7, 600.00, "Reabastecimiento"),
    -> (37, 9, 12, 2, 1200.00, "Para reparación de maquinaria"),
    -> (38, 9, 13, 8, 1800.00, "Requisito de proyecto"),
    -> (39, 10, 14, 5, 450.00, "Compra mensual"),
    -> (40, 10, 15, 3, 300.00, "Urgente para cliente"),
    -> (41, 1, 16, 4, 600.00, "Para proyecto de construcción"),
    -> (42, 1, 17, 2, 1200.00, "Urgente para entrega"),
    -> (43, 2, 18, 7, 250.00, "Reabastecimiento"),
    -> (44, 2, 19, 8, 1500.00, "Para cliente especial"),
    -> (45, 3, 20, 6, 300.00, "Necesidad de última hora"),
    -> (46, 3, 21, 4, 600.00, "Compra por demanda"),
    -> (47, 4, 22, 3, 1200.00, "Proyecto de construcción"),
    -> (48, 4, 23, 2, 1800.00, "Reabastecimiento urgente"),
    -> (49, 5, 24, 1, 450.00, "Para uso inmediato"),
    -> (50, 5, 25, 5, 300.00, "Reemplazo de herramientas viejas");
Query OK, 50 rows affected (0.003 sec)
Records: 50  Duplicates: 0  Warnings: 0
```

Listing 16: Reporte de compras del mes de enero que incluya el nombre del proveedor el nombre de la herramienta cantidad precio unitario y precio total

```
SELECT
MONTH(p.date) AS Mes,
 s.name AS "Nombre del proveedor",
 t.name AS "Nombre de la herramienta",
 b.amount AS "Cantidad",
 b.unitPrice AS "Precio unitario",
 b.amount * b.unitPrice AS "Precio total"
FROM purchase AS p
LEFT JOIN
 supplier AS s
ON p.idSupplier = s.idSupplier
LEFT JOIN
 buyTool AS b
ON p.idPurchase = b.idPurchase
LEFT JOIN
 inventory AS i
ON b.idInventory = i.idInventory
```

```
LEFT JOIN
 tool AS t
ON i.idTool = t.idTool
HAVING Mes = 01;
```

$$\pi_{\text{Mes, Nombre del proveedor, Nombre de la herramienta, Cantidad, Precio unitario, Precio total}} \left(\sigma_{\text{Mes=01}} \left(\tau_{\text{MONTH}(p.date)} \times ((p \bowtie_{p.idSupplier=s.idSupplier} s) \bowtie_{p.idPurchase=b.idPurchase} b \bowtie_{b.idInventory=i.idInventory} i \bowtie_{i.idTool=t.idTool} t)\right)\right) \tag{3}$$

Mes	Nombre del proveedor	Nombre de la herramienta	Cantidad	Precio unitario	Precio total
1	Herramientas del Norte	Taladro Inalámbrico	5	600.00	3000.00
1	Herramientas del Norte	Sierra Circular	3	1200.00	3600.00
1	Herramientas del Norte	Taladro Inalámbrico	4	600.00	2400.00
1	Herramientas del Norte	Sierra Circular	7	1200.00	8400.00
1	Herramientas del Norte	Taladro Inalámbrico	4	600.00	2400.00
1	Herramientas del Norte	Sierra Circular	2	1200.00	2400.00
1	Proveedores de Herramientas S.A.	Juego de Destornilladores	10	250.00	2500.00
1	Proveedores de Herramientas S.A.	Martillo de Framing	2	1500.00	3000.00
1	Proveedores de Herramientas S.A.	Juego de Destornilladores	8	250.00	2000.00
1	Proveedores de Herramientas S.A.	Martillo de Framing	6	1500.00	9000.00
1	Proveedores de Herramientas S.A.	Juego de Destornilladores	7	250.00	1750.00
1	Proveedores de Herramientas S.A.	Martillo de Framing	8	1500.00	12000.00
1	Distribuciones de Herramientas	Llave de Ajuste	7	300.00	2100.00
1	Distribuciones de Herramientas	Taladro Inalámbrico	5	600.00	3000.00
1	Distribuciones de Herramientas	Llave de Ajuste	2	300.00	600.00
1	Distribuciones de Herramientas	Taladro Inalámbrico	9	600.00	5400.00
1	Distribuciones de Herramientas	Llave de Ajuste	6	300.00	1800.00
1	Distribuciones de Herramientas	Taladro Inalámbrico	4	600.00	2400.00

18 rows in set (0.001 sec)

Listing 17: – Reporte de inventario de la bodega de la calle 15 que incluya el nombre de la herramienta cantidad y costo total (precio de compra * cantidad)

```
SELECT
w.street AS "Calle de bodega",
t.name AS "Nombre de la herramienta",
i.amount AS "Cantidad",
i.amount * i.purchasePrice AS "Costo total"
FROM warehouse AS w
LEFT JOIN
 inventory AS i
ON w.idWarehouse = i.idWarehouse
LEFT JOIN
 tool AS t
ON i.idTool = t.idTool
WHERE w.street = "Calle 15";
```

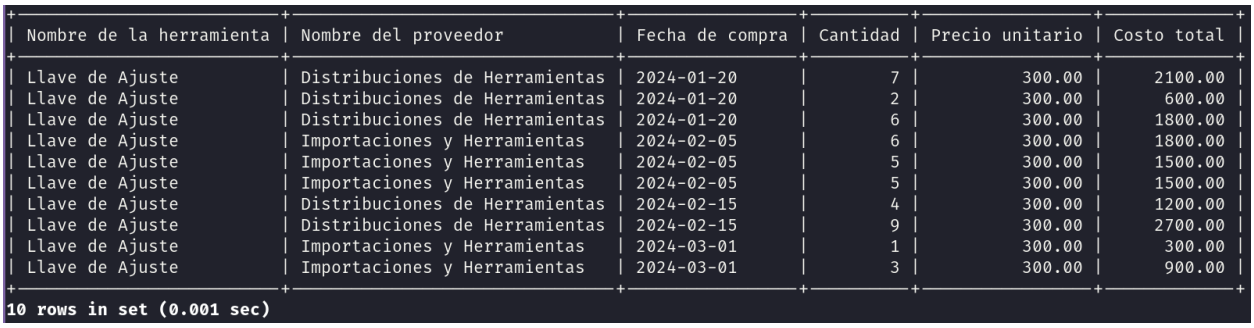
$$\pi_{\text{Calle de bodega, Nombre de la herramienta, Cantidad, Costo total}} \left(\sigma_{w.street=\text{"Calle 15"}} \left((w \bowtie_{w.idWarehouse=i.idWarehouse} i) \bowtie_{i.idTool=t.idTo} \right.\right. \tag{4}$$

```
MariaDB [distribuidora]> SELECT
    ->     w.street AS "Calle de bodega",
    ->     t.name AS "Nombre de la herramienta",
    ->     i.amount AS "Cantidad",
    ->     i.amount * i.purchasePrice AS "Costo total"
    -> FROM warehouse AS w
    -> LEFT JOIN
    ->     inventory AS i
    ->     ON w.idWarehouse = i.idWarehouse
    -> LEFT JOIN
    ->     tool AS t
    ->     ON i.idTool = t.idTool
    -> WHERE w.street = "Calle 15";
+-----------------+-----------------------------+-----------+--------------+
| Calle de bodega | Nombre de la herramienta    | Cantidad  | Costo total  |
+-----------------+-----------------------------+-----------+--------------+
| Calle 15        | Taladro Inalámbrico         |        22 |     13200.00 |
| Calle 15        | Sierra Circular             |        18 |     21600.00 |
| Calle 15        | Juego de Destornilladores   |        28 |      7000.00 |
| Calle 15        | Martillo de Framing         |        12 |     18000.00 |
| Calle 15        | Llave de Ajuste             |        35 |     10500.00 |
+-----------------+-----------------------------+-----------+--------------+
5 rows in set (0.001 sec)
```

Listing 18: Reporte de compras de la herramienta “Llave de Ajuste” durante el año 2024 el nombre del proveedor fecha de compra cantidad precio unitario y costo total.

```
SELECT
t.name AS "Nombre de la herramienta",
s.name AS "Nombre del proveedor",
p.date AS "Fecha de compra",
b.amount AS "Cantidad",
b.unitPrice AS "Precio unitario",
b.amount * b.unitPrice AS "Costo total"
FROM purchase AS p
LEFT JOIN
supplier AS s
ON p.idSupplier = s.idSupplier
LEFT JOIN
buyTool AS b
ON p.idPurchase = b.idPurchase
LEFT JOIN
inventory AS i
ON b.idInventory = i.idInventory
LEFT JOIN
tool AS t
ON i.idTool = t.idTool
HAVING YEAR(p.date) = 2024
AND t.name = "Llave de Ajuste";
```

$$\pi_{\text{Nombre de la herramienta, Nombre del proveedor, Fecha de compra, Cantidad, Precio unitario, Costo total}} \left(\sigma_{\text{YEAR}(p.date)=2024 \wedge t.name=\text{"Llave de A}}\right) \quad (5)$$



Nombre de la herramienta	Nombre del proveedor	Fecha de compra	Cantidad	Precio unitario	Costo total
Llave de Ajuste	Distribuciones de Herramientas	2024-01-20	7	300.00	2100.00
Llave de Ajuste	Distribuciones de Herramientas	2024-01-20	2	300.00	600.00
Llave de Ajuste	Distribuciones de Herramientas	2024-01-20	6	300.00	1800.00
Llave de Ajuste	Importaciones y Herramientas	2024-02-05	6	300.00	1800.00
Llave de Ajuste	Importaciones y Herramientas	2024-02-05	5	300.00	1500.00
Llave de Ajuste	Importaciones y Herramientas	2024-02-05	5	300.00	1500.00
Llave de Ajuste	Distribuciones de Herramientas	2024-02-15	4	300.00	1200.00
Llave de Ajuste	Distribuciones de Herramientas	2024-02-15	9	300.00	2700.00
Llave de Ajuste	Importaciones y Herramientas	2024-03-01	1	300.00	300.00
Llave de Ajuste	Importaciones y Herramientas	2024-03-01	3	300.00	900.00

10 rows in set (0.001 sec)

Listing 19: Listado de responsables de las bodegas de la empresa con calle número y teléfono.

```
SELECT
w.manager AS "Responsable de bodega",
w.street AS "Calle",
w.number AS "Número",
w.phone AS "Teléfono"
FROM warehouse AS w;
```

$$\pi_{\text{Responsable de bodega, Calle, Número, Teléfono}}(w) \tag{6}$$

```
MariaDB [distribuidora]> SELECT
    ->     w.manager AS "Responsable de bodega",
    ->     w.street AS "Calle",
    ->     w.number AS "Número",
    ->     w.phone AS "Teléfono"
    -> FROM warehouse AS w;
+-------------------------+-------------------+---------+-----------------+
| Responsable de bodega   | Calle             | Número  | Teléfono        |
+-------------------------+-------------------+---------+-----------------+
| Ricardo Gómez           | Av. Juárez        | 100     | 555-1001-100    |
| Patricia Ramírez        | Calle 15          | 200     | 333-1002-200    |
| Fernando Torres         | Calle 10          | 300     | 818-1003-300    |
| Isabel Castro           | Av. Insurgentes   | 400     | 222-1004-400    |
| Javier Martínez         | Calle 5           | 500     | 664-1005-500    |
+-------------------------+-------------------+---------+-----------------+
5 rows in set (0.001 sec)
```

Listing 20: Listado de contactos con los proveedores con nombre de contacto nombre de proveedor teléfono y correo electrónico.

```
SELECT
s.contact AS "Nombre de contacto",
s.name AS "Proveedor",
s.phone AS "Teléfono",
s.email AS "Correo electrónico"
FROM supplier AS s;
```

$$\pi_{\text{Nombre de contacto, Proveedor, Teléfono, Correo electrónico}}(s) \tag{7}$$

```
MariaDB [distribuidora]> SELECT
    ->     s.contact AS "Nombre de contacto",
    ->     s.name AS "Proveedor",
    ->     s.phone AS "Teléfono",
    ->     s.email AS "Correo electrónico"
    -> FROM supplier AS s;
+---------------------+-----------------------------------+--------------+-----------------------------------------+
| Nombre de contacto  | Proveedor                         | Teléfono     | Correo electrónico                      |
+---------------------+-----------------------------------+--------------+-----------------------------------------+
| Alberto Ruiz        | Herramientas del Norte            | 818-123-4567 | ventas@herramientasnorte.com            |
| Sofía Gómez         | Proveedores de Herramientas S.A.  | 555-234-5678 | contactoprovherr@hotmail.com            |
| Carlos Mendoza      | Distribuciones de Herramientas    | 333-345-6789 | infodistribucionesherramientas@gmail.com |
| María Elena Pérez   | Soluciones en Herramientas        | 222-456-7890 | soporte@solucionesherramientas.com      |
| Luis Miguel Torres  | Importaciones y Herramientas      | 664-567-8901 | ventas@toolimport.com                   |
+---------------------+-----------------------------------+--------------+-----------------------------------------+
5 rows in set (0.001 sec)
```

Listing 21: Reporte de herramientas compradas cuyo precio unitario se menor o igual a $250.00 que incluya nombre de la herramienta fecha de compra y cantidad comprada en orden cronológico descendiente.

```
SELECT
t.name AS "Nombre de la herramienta",
b.unitPrice AS "Precio unitario",
p.date AS "Fecha de compra",
b.amount AS "Cantidad comprada"
FROM purchase AS p
LEFT JOIN
buyTool AS b
ON p.idPurchase = b.idPurchase
LEFT JOIN
inventory AS i
ON b.idInventory = i.idInventory
LEFT JOIN
tool AS t
ON i.idTool = t.idTool
WHERE b.unitPrice <= 250.00
ORDER BY p.date DESC;
```

$$\pi_{\text{Nombre de la herramienta, Precio unitario, Fecha de compra, Cantidad comprada}} (\sigma_{b.unitPrice \leq 250.00} ((p \bowtie_{p.idPurchase = b.idPurchase} b) \bowtie_{b.idIn} \tag{8}$$

```
MariaDB [distribuidora]> SELECT
    ->     t.name AS "Nombre de la herramienta",
    ->     b.unitPrice AS "Precio unitario",
    ->     p.date AS "Fecha de compra",
    ->     b.amount AS "Cantidad comprada"
    -> FROM purchase AS p
    -> LEFT JOIN
    ->     buyTool AS b
    ->     ON p.idPurchase = b.idPurchase
    -> LEFT JOIN
    ->     inventory AS i
    ->     ON b.idInventory = i.idInventory
    -> LEFT JOIN
    ->     tool AS t
    ->     ON i.idTool = t.idTool
    -> WHERE b.unitPrice ⩽ 250.00
    -> ORDER BY p.date DESC;
+-----------------------------+-----------------+-----------------+--------------------+
| Nombre de la herramienta    | Precio unitario | Fecha de compra | Cantidad comprada  |
+-----------------------------+-----------------+-----------------+--------------------+
| Juego de Destornilladores   |          250.00 | 2024-02-12      |                  1 |
| Juego de Destornilladores   |          250.00 | 2024-02-12      |                  5 |
| Juego de Destornilladores   |          250.00 | 2024-01-18      |                 10 |
| Juego de Destornilladores   |          250.00 | 2024-01-18      |                  8 |
| Juego de Destornilladores   |          250.00 | 2024-01-18      |                  7 |
+-----------------------------+-----------------+-----------------+--------------------+
5 rows in set (0.004 sec)
```

Listing 22: Reporte de herramientas en el inventario cuyo stock sea entre 5 y 20 piezas que incluya calle y número de la bodega nombre de la herramienta ubicación y cantidad en existencia.

```
SELECT
w.street AS "Calle",
w.number AS "Número",
t.name AS "Nombre de la herramienta",
i.location AS "Ubicación",
i.amount AS Cantidad
FROM inventory AS i
LEFT JOIN
warehouse AS w
ON w.idWarehouse = i.idWarehouse
LEFT JOIN
tool AS t
ON i.idTool = t.idTool
HAVING Cantidad BETWEEN 5 AND 20;
```

$$\pi_{\text{Calle, Número, Nombre de la herramienta, Ubicación, Cantidad}} \left(\sigma_{5 \leq i.amount \leq 20} \left((i \bowtie_{i.idWarehouse = w.idWarehouse} w) \bowtie_{i.idTool = t.idTool} t \right)\right) \tag{9}$$

Calle	Número	Nombre de la herramienta	Ubicación	Cantidad
Av. Juárez	100	Sierra Circular	Ubicación A2	20
Av. Juárez	100	Juego de Destornilladores	Ubicación B1	15
Av. Juárez	100	Martillo de Framing	Ubicación B2	10
Calle 15	200	Sierra Circular	Ubicación A4	18
Calle 15	200	Martillo de Framing	Ubicación B4	12
Calle 10	300	Taladro Inalámbrico	Ubicación D1	10
Calle 10	300	Sierra Circular	Ubicación D2	20
Calle 10	300	Juego de Destornilladores	Ubicación E1	15
Calle 10	300	Martillo de Framing	Ubicación E2	8
Calle 10	300	Llave de Ajuste	Ubicación F1	12
Av. Insurgentes	400	Sierra Circular	Ubicación C3	10
Av. Insurgentes	400	Juego de Destornilladores	Ubicación D3	18
Av. Insurgentes	400	Martillo de Framing	Ubicación F2	5
Calle 5	500	Sierra Circular	Ubicación D4	12
Calle 5	500	Juego de Destornilladores	Ubicación F3	15
Calle 5	500	Martillo de Framing	Ubicación E4	10

16 rows in set (0.002 sec)

Listing 23: Reporte del stock de todas las bodegas que incluya calle número responsable teléfono y total de herramientas almacenadas.

```
SELECT
w.idWarehouse AS "ID de bodega",
w.street AS "Calle",
w.number AS "Número",
w.manager AS "Responsable",
w.phone AS "Teléfono",
SUM(i.amount) AS "Total de herramientas"
FROM warehouse AS w
LEFT JOIN
inventory AS i
ON w.idWarehouse = i.idWarehouse
GROUP BY w.idWarehouse;
```

$$\pi_{\text{ID de bodega, Calle, Número, Responsable, Teléfono, Total de herramientas}} \left(\gamma_{w.idWarehouse, w.street, w.number, w.manager, w.phone, \text{SUM}(i.amount} \right) \quad (10)$$

```
MariaDB [distribuidora]> SELECT
    ->     w.idWarehouse AS "ID de bodega",
    ->     w.street AS "Calle",
    ->     w.number AS "Número",
    ->     w.manager AS "Responsable",
    ->     w.phone AS "Teléfono",
    ->     SUM(i.amount) AS "Total de herramientas"
    -> FROM warehouse AS w
    -> LEFT JOIN
    ->     inventory AS i
    ->     ON w.idWarehouse = i.idWarehouse
    -> GROUP BY w.idWarehouse;
+---------------+-----------------+---------+-------------------+---------------+--------------------------+
| ID de bodega | Calle           | Número  | Responsable       | Teléfono      | Total de herramientas |
+---------------+-----------------+---------+-------------------+---------------+--------------------------+
|            1 | Av. Juárez      | 100     | Ricardo Gómez     | 555-1001-100 |                      100 |
|            2 | Calle 15        | 200     | Patricia Ramírez  | 333-1002-200 |                      115 |
|            3 | Calle 10        | 300     | Fernando Torres   | 818-1003-300 |                       65 |
|            4 | Av. Insurgentes | 400     | Isabel Castro     | 222-1004-400 |                       88 |
|            5 | Calle 5         | 500     | Javier Martínez   | 664-1005-500 |                      100 |
+---------------+-----------------+---------+-------------------+---------------+--------------------------+
5 rows in set (0.001 sec)
```

Listing 24: Reporte de valor de inventario de todas las bodegas que incluya calle estado y monto total de precio de venta de las herramientas almacenadas.

```
SELECT
w.idWarehouse AS "ID de bodega",
w.street AS "Calle",
w.state AS "Estado",
SUM(b.unitPrice) AS "Monto total de precio de venta"
FROM warehouse AS w
LEFT JOIN
inventory AS i
ON w.idWarehouse = i.idWarehouse
LEFT JOIN
buyTool AS b
ON b.idInventory = i.idInventory
GROUP BY w.idWarehouse;
```

$$\pi_{\text{ID de bodega, Calle, Estado, Monto total de precio de venta}}\left(\gamma_{w.idWarehouse,w.street,w.state,\text{SUM}(b.unitPrice)\rightarrow\text{Monto total de precio de venta}}\left((\right.\right. \tag{11}$$

```
MariaDB [distribuidora]> SELECT
    ->     w.idWarehouse AS "ID de bodega",
    ->     w.street AS "Calle",
    ->     w.state AS "Estado",
    ->     SUM(b.unitPrice) AS "Monto total de precio de venta"
    -> FROM warehouse AS w
    -> LEFT JOIN
    ->     inventory AS i
    ->     ON w.idWarehouse = i.idWarehouse
    -> LEFT JOIN
    ->     buyTool AS b
    ->     ON b.idInventory = i.idInventory
    -> GROUP BY w.idWarehouse;
+---------------+-----------------+-----------------+-----------------------------------+
| ID de bodega | Calle           | Estado          | Monto total de precio de venta |
+---------------+-----------------+-----------------+-----------------------------------+
|            1 | Av. Juárez      | CDMX            |                           8200.00 |
|            2 | Calle 15        | Jalisco         |                           8200.00 |
|            3 | Calle 10        | Nuevo León      |                           8200.00 |
|            4 | Av. Insurgentes | Puebla          |                           8200.00 |
|            5 | Calle 5         | Baja California |                           8200.00 |
+---------------+-----------------+-----------------+-----------------------------------+
5 rows in set (0.001 sec)
```

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

Castillo, R. (2020). “¿Qué es inyección SQL?”.

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.

Silberschatz, A. et al. (2002). *Fundamentos de bases de datos* (Vol. 11). McGraw-Hill Ciudad de México, México.

Zambrano Ramírez, R. (2008). “Base de datos relacionales”.