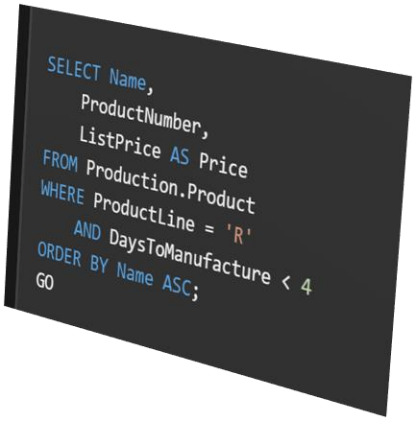
5. SQL – Structured Query Language



```
SELECT Name,
    ProductNumber,
    ListPrice AS Price
FROM Production.Product
WHERE ProductLine = 'R'
    AND DaysToManufacture < 4
ORDER BY Name ASC;
GO
```

Cuando se tiene ya un diseño estructurado de una base de datos, dicha estructura está preparado para recibir los datos y almacenarlos. Estos pueden provenir de diferentes fuentes, pero en última instancia quien recibe los datos es un sistema de gestión de bases de datos o DBMS (Daba base management system), también conocido como motor de base de datos; es una herramienta tecnológica que provee dos sub lenguajes: DDL y DML incorporados actualmente con un lenguaje genérico llamado SQL (Lenguaje estructurado de consulta).

El SQL, es un lenguaje de programación para almacenar y procesar información en una base de datos relacional. La información se almacena en forma de tabla, con filas y columnas que representan diferentes atributos de datos y las diversas relaciones entre los valores de datos. El SQL ofrece instrucciones para realizar operaciones CRUD (Create, Read, Update, Delete) sobre una base de datos; También ofrece instrucciones para crear las estructuras, y gestionar la base de datos.

Los tipos de datos y algunas pequeñas diferencias de sintaxis se tienen al comparar diferentes DBMS, pero conservan la raíz del estándar ISO/IEC 9075, que formaliza las estructuras y comportamientos de la sintaxis SQL en todos los productos de bases de datos.

5.1 Importancia de SQL

EL SQL es el corazón de un DBMS, porque además de proveer herramientas para definición y manipulación se usa para tares primordiales como:

- **Interoperabilidad:** Es un lenguaje estándar adoptado por la mayoría de los sistemas de gestión de bases de datos relacionales (RDBMS).
- **Análisis de datos**: Permite realizar análisis complejos y generación de datos estructurados informes y dataset.
- **Automatización**: Se realiza a través de procedimientos almacenados y triggers.
- **Seguridad**: SQL ayuda a controlar acceso a los datos y mantener la integridad de los datos.

5.2 DDL (Data Definition Language -Lenguaje de Definición de Datos)

Es un sub-lenguaje de SQL que se utiliza para definir y gestionar la estructura de las bases de datos y sus objetos, como tablas, índices y vistas. Las operaciones DDL afectan la estructura de la base de datos, no los datos en sí. El DDL tiene instrucciones propias para las operaciones en una base de datos.

Ejemplos de DDL:

```
/*==============================================================*/
/* Table: Formadores                                            */
/*==============================================================*/
create table Formadores (
   Cod_formador         integer              identity,
   Nombre_empresa       char(60)             null,
   Nit                  char(20)             null,
   Telefono             char(25)             null,
   Web                  char(60)             null,
   email                char(60)             null,
   contacto             char(60)             null,
   Descripcion          char(200)            null,
   constraint PK_FORMADORES primary key (Cod_formador)
)
```

Formadores		
Cod_formador	integer	<pk>
Nombre_empresa	char(60)	
Nit	char(20)	
Telefono	char(25)	
Web	char(60)	
email	char(60)	
contacto	char(60)	
Descripcion	char(200)	

1. **CREATE TABLE:** Crea una nueva tabla en la base de datos.

```
CREATE TABLE empleados (
    id INT PRIMARY KEY,
    nombre VARCHAR(100),
    puesto VARCHAR(50),
    salario DECIMAL(10, 2)   );
```

2. **ALTER TABLE:** Modifica la estructura de una tabla existente.

   ```
   ALTER TABLE empleados
   ADD fecha_contratacion DATE;
   ```

3. **DROP TABLE:** Elimina una tabla existente de la base de datos.

   ```
   DROP TABLE empleados;
   ```

4. **CREATE INDEX:** Crea un índice en una columna de una tabla para mejorar la velocidad de las consultas.

   ```
   CREATE INDEX idx_nombre ON empleados (nombre);
   ```

5.3 DML (Data Manipulation Language – Lenguaje de Manipulación de datos)

El DML es un sub-lenguaje de SQL que se utiliza para gestionar los datos dentro de las estructuras definidas por DDL. Las operaciones DML pueden afectar los datos contenidos en las tablas de la base de datos.

Ejemplos de DML

```
INSERT INTO conductores
     ( doc_id,
       nombre,
       vehiculo,
       descripcion,
       telefono )
VALUES ( '2323',
       "Juan valdez",
       "bxb.23e",
       "Por cnacelar pago",
       "3122222c2" ) ;
```

conductores	Type
doc_id	char(20)
nombre	char(50)
vehiculo	char(200)
descripcion	text
telefono	char(50)

1. **INSERT:** Inserta nuevos datos en una tabla.

   ```
   INSERT INTO empleados (id, nombre, puesto, salario)
   VALUES (1, 'Juan Pérez', 'Desarrollador', 60000);
   ```

2. **SELECT:** Recupera datos de la base de datos.

   ```
   SELECT * FROM empleados WHERE puesto = 'Desarrollador';
   ```

3. **UPDATE:** Actualiza datos existentes en una tabla.

```
UPDATE empleados
SET salario = 65000
WHERE nombre = 'Juan Pérez';
```

4. **DELETE:** Elimina datos de una tabla.

```
DELETE FROM empleados
WHERE nombre = 'Juan Pérez';
```

5.4 **Funciones más comunes de SQL**

Se agrupan en 3 grupos funciones DDL, DML y DCL (Data Languaje Control)

5.4.1. **DDL** (Data Definition Language)

- CREATE TABLE: Crea una nueva tabla.
- ALTER TABLE: Modifica una tabla existente.
- DROP TABLE: Elimina una tabla.

5.4.2. **DML** (Data Manipulation Language)

- SELECT: Recupera datos de la base de datos.
- INSERT: Inserta nuevos datos.
- UPDATE: Actualiza datos existentes.
- DELETE: Elimina datos.

5.4.3. **DCL** (Data Control Language): Controla el acceso a los datos.

GRANT: Concede permisos.

La instrucción GRANT en SQL se utiliza para otorgar permisos específicos a usuarios o roles sobre objetos de la base de datos. Estos permisos determinan qué operaciones pueden realizar los usuarios en esos objetos, como por ejemplo: permite eliminar registros de una tabla.

REVOKE: Revoca permisos.

Se utiliza para revocar permisos específicos que se han otorgado previamente a usuarios o roles sobre objetos de la base de datos.

Ejemplo: GRANT

Supongamos que tenemos una base de datos llamada IIR2 y una tabla llamada jugadores. Queremos otorgar a un usuario llamado juan el permiso de SELECT sobre la tabla jugadores. Para ello, podemos utilizar la siguiente instrucción GRANT:

```
GRANT SELECT ON jugadores TO juan;
```

Explicación:
GRANT: Es la palabra clave que indica que se va a otorgar un permiso.
SELECT: Especifica el tipo de permiso que se va a otorgar. En este caso, estamos otorgando el permiso de SELECT.
ON jugadores: Indica el objeto sobre el que se va a otorgar el permiso. En este caso, el objeto es la tabla jugadores.
TO juan: Indica el usuario o rol al que se le va a otorgar el permiso. En este caso, el usuario es juan.

Ejemplo REVOKE

Supongamos que la tabla jugadores de la base de datos IIR2 tiene otorgado el permiso de SELECT al usuario juan. Para revocar este permiso, podemos utilizar la siguiente instrucción:

```
REVOKE SELECT ON jugadores FROM juan;
```

Explicación:
REVOKE: Es la palabra clave que indica que se va a revocar un permiso.

SELECT: Especifica el tipo de permiso que se va a revocar. En este caso, estamos revocando el permiso de SELECT.
ON jugadores: Indica el objeto sobre el que se va a revocar el permiso. En este caso, el objeto es la tabla jugadores.
FROM juan: Indica el usuario o rol al que se le va a revocar el permiso. En este caso, el usuario es juan.

5.4.4. **TCL** (Transaction Control Language): Administra las transacciones.

COMMIT: Confirma una transacción.

Se utiliza para marcar el final de una transacción y hacer que los cambios realizados en la base de datos durante esa transacción sean permanentes.
En otras palabras, COMMIT convierte los cambios temporales en cambios permanentes.

ROLLBACK: Deshace una transacción.

Se utiliza para revertir los cambios realizados en la base de datos durante una transacción. ROLLBACK deshace las operaciones que se han ejecutado dentro de una transacción, dejando la base de datos en el estado que tenía antes de iniciar la transacción.

Un ROLLBACK se usa en las siguientes situaciones:

- Cuando se produce un error durante una transacción para evitar que la base de datos quede en un estado inconsistente.
- Cuando se decide cancelar una transacción, se decide no realizar los cambios, y dejar la base de datos en su estado original.

¿Qué es una transacción?

Una transacción es una unidad lógica de trabajo dentro de una base de datos. Agrupa una serie de operaciones de SQL que se ejecutan como una sola unidad. Las transacciones tienen las propiedades de atomicidad, consistencia, aislamiento y durabilidad. El COMMIT debe utilizar al final de cada transacción para garantizar que los cambios realizados sean permanentes.

Ejemplo: COMMIT
Supongamos que queremos actualizar el nombre y el apellido de un jugador en la tabla jugadores de la base de datos IIR2.

```
BEGIN TRANSACTION;
UPDATE jugadores
SET nombre = 'Juan', apellido = 'Pérez'
WHERE idJugador = 10;
COMMIT;
```

Explicación:
BEGIN TRANSACTION; Esta instrucción inicia una nueva transacción.
UPDATE jugadores...: Esta instrucción actualiza el nombre y el apellido del jugador con el ID 10.
COMMIT; Esta instrucción confirma la transacción y hace que los cambios realizados sean permanentes.
Si no hubiéramos utilizado la instrucción COMMIT, los cambios realizados en la tabla jugadores no se habrían guardado de forma permanente.

Ejemplo de ROLLBACK
Supongamos que queremos actualizar el nombre y el apellido de un jugador en la tabla jugadores de la base de datos IIR2. Para ello, iniciamos una transacción y ejecutamos la siguiente instrucción SQL:

```
SQL
BEGIN TRANSACTION;
UPDATE jugadores
SET nombre = 'Juan', apellido = 'Pérez'
WHERE idJugador = 10;
```

Sin embargo, antes de confirmar la transacción, nos damos cuenta de que hemos introducido un error en el apellido y queremos corregirlo.
--En este caso, podemos utilizar ROLLBACK para revertir los cambios realizados en la tabla jugadores:

```
ROLLBACK;
```

Explicación:
ROLLBACK; Esta instrucción revierte todos los cambios realizados en la tabla jugadores durante la transacción actual.
Como resultado de ejecutar ROLLBACK, el nombre y el apellido del jugador con el ID 10 no se habrán actualizado.

5.5 **Consultas CRUD**

La cláusula WHERE en SQL se utiliza para filtrar registros de una tabla en función de una condición específica. Esto permite seleccionar solo aquellos registros que cumplen con los criterios definidos.

CRUD es un acrónimo que se refiere a las cuatro operaciones básicas de manipulación de datos en una base de datos:

La clausula WHERE se utiliza para filtrar la información que se requiere y para realizar las operaciones de unión entre tablas (Join)

Create (INSERT):	INSERT INTO empleados (nombre, puesto, salario) VALUES ('Juan Pérez', 'Desarrollador', 60000);
Read (SELECT):	SELECT * FROM empleados WHERE puesto = 'Desarrollador';
Update (UPDATE)	UPDATE empleados SET salario = 65000 WHERE nombre = 'Juan Pérez';
Delete (DELETE)	DELETE FROM empleados WHERE nombre = 'Juan Pérez';

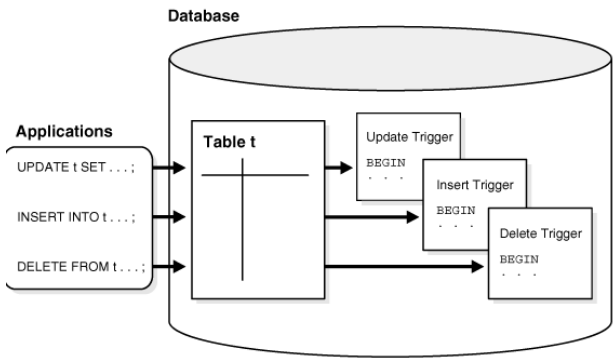
5.6 Procedimientos Almacenados

Un procedimiento almacenado es un conjunto de instrucciones (algoritmo o proceso) SQL que se puede guardar en el DBMS y reutilizar. Ayuda a encapsular operaciones complejas y a mejorar el rendimiento y la seguridad.

```
CREATE PROCEDURE ActualizarSalario
  @EmpleadoID INT,
  @NuevoSalario DECIMAL(10, 2)
AS
BEGIN
  UPDATE empleados SET salario = @NuevoSalario WHERE id =
@EmpleadoID;
END;
```

5.7 Triggers

Un trigger es un conjunto de instrucciones SQL que se ejecuta automáticamente en respuesta a ciertos eventos en una tabla o vista, como `INSERT`, `UPDATE` o `DELETE`.




```
CREATE TRIGGER trg_AntesDeInsertar
ON empleados
INSTEAD OF INSERT
AS
BEGIN
  DECLARE @salario DECIMAL(10, 2);
  SELECT @salario = salario FROM INSERTED;
  IF @salario < 30000
    BEGIN
      RAISERROR ('El salario no puede ser menor a 30000', 16,
1);
      ROLLBACK;
    END
  ELSE
    BEGIN
      INSERT INTO empleados (nombre, puesto, salario)
      SELECT nombre, puesto, salario FROM INSERTED;
    END
END;
```

5.8 Funciones de Agregación

Las funciones de agregación son procedimientos ya establecidos en el DBMS para realizar cálculos sobre un conjunto de datos y devolver resultados sobre estos. Son muy útiles en procesos de ciencia de datos y analitica.

Función	Descripción	Ejemplo
COUNT	Cuenta el número de filas en una tabla o el número de valores no nulos en una columna.	SELECT COUNT(*) FROM clientes; SELECT COUNT(nombre) FROM clientes WHERE ciudad = 'Medellín';
AVG	Calcula el promedio de los valores numéricos en una columna.	SELECT AVG(salario) FROM empleados;
SUM	Calcula la suma de los valores numéricos en una columna.	SELECT SUM(ventas) FROM productos;
MIN	Devuelve el valor mínimo en una columna.	SELECT MIN(precio) FROM productos;
MAX	Devuelve el valor máximo en una columna.	SELECT MAX(puntuacion) FROM valoraciones;
GROUP BY	Agrupa las filas de una tabla en función de una o más columnas y aplica las funciones de agregación a cada grupo.	SELECT ciudad, COUNT(*) AS clientes FROM clientes GROUP BY ciudad;
HAVING	Filtra los grupos de filas resultantes de una consulta GROUP BY.	SELECT ciudad, COUNT(*) AS clientes FROM clientes GROUP BY ciudad HAVING clientes > 100;
DISTINCT	Elimina los valores duplicados en una columna.	SELECT DISTINCT ciudad FROM clientes;
COUNT DISTINCT	Cuenta el número de valores únicos en una columna.	SELECT COUNT(DISTINCT ciudad) AS ciudades_distintas FROM clientes;
STDDEV	Calcula la desviación estándar de los valores numéricos en una columna.	SELECT STDDEV(salario) FROM empleados;
VARIANCE	Calcula la varianza de los valores numéricos en una columna.	SELECT VARIANCE(salario) FROM empleados;
KURTOSIS	Calcula la kurtosis de los valores numéricos en una columna.	SELECT KURTOSIS(salario) FROM empleados;
SKEWNESS	Calcula la asimetría de los valores numéricos en una columna.	SELECT SKEWNESS(salario) FROM empleados;

5.9 SQL en Ciencia de Datos

En el ámbito de la ciencia de datos, SQL es crucial para la preparación de datos, limpieza y análisis exploratorio de datos. Las consultas SQL permiten extraer conjuntos de datos relevantes, agregar datos y realizar un análisis inicial antes de aplicar técnicas más avanzadas de análisis de datos y aprendizaje automático.

SQL es una herramienta indispensable para los científicos de datos e ingenieros de IA, ya que les permite acceder, manipular, analizar y compartir grandes volúmenes de datos de manera eficiente y escalable. Su integración con herramientas populares de Ciencia de Datos e IA, junto con su capacidad para mantener y actualizar datos, lo convierten en un lenguaje aliado para el desarrollo de modelos de IA confiables y efectivos. A continuación, se presentan algunas razones por las que SQL es importante en Ciencia de Datos e IA.

Acceso y manipulación de altos volúmenes de datos

Las bases de datos relacionales almacenan grandes cantidades de datos estructurados, esenciales para el entrenamiento y desarrollo de modelos de IA. SQL permite a los científicos de datos acceder, extraer, filtrar, transformar y agregar estos datos de manera eficiente, lo que facilita su análisis y preparación para el modelado.

Integración con herramientas de Ciencia de Datos e IA

SQL se integran herramientas y frameworks populares en Ciencia de Datos e IA, como Python, R, TensorFlow y PyTorch. Esto permite importar datos desde bases de datos relacionales a estas herramientas para su análisis y modelado, y luego exportar los resultados de vuelta a las bases de datos para su almacenamiento y consulta.

Escalabilidad y rendimiento

Las bases de datos relacionales y SQL están diseñados para manejar grandes volúmenes de datos de manera eficiente. Importante para la IA, donde los modelos de aprendizaje automático a menudo se entrenan con conjuntos de datos masivos. SQL permite a los científicos de datos optimizar el proceso de entrenamiento y desarrollo de modelos.

Colaboración y compartición de datos

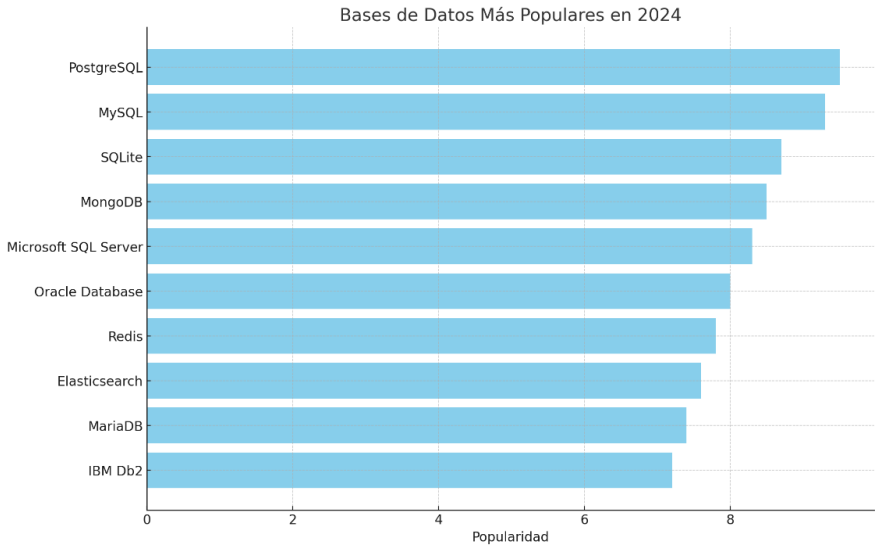
SQL facilita la colaboración entre científicos de datos, permite compartir consultas y resultados de análisis de forma estandarizada. Esto promueve la transparencia y reproducibilidad de los procesos.

Mantenimiento y actualización de datos

Con el SQL se permite a los científicos de datos mantener y actualizar los datos en las bases de datos relacionales. Esto es importante para garantizar que los modelos de IA se entrenen con datos precisos y actualizados, lo que mejora su rendimiento y confiabilidad.

5.10 COMAPARATIVO DE ALGUNOS DBMS

Característica	SQL Server	Oracle	MySQL	PostgreSQL
Licencia	Comercial y de código abierto	Comercial y de código abierto	Código abierto	Código abierto
Costo	Alto	Alto	Bajo	Bajo
Escalabilidad	Muy alta	Muy alta	Alta	Alta
Seguridad	Muy alta	Muy alta	Alta	Alta
Facilidad de uso	Media	Media	Alta	Alta
Soporte para la comunidad	Grande	Grande	Muy grande	Muy grande
Casos de uso	Grandes empresas, aplicaciones empresariales críticas	Grandes empresas, aplicaciones empresariales críticas	Pequeñas y medianas empresas, aplicaciones web	Pequeñas y medianas empresas, aplicaciones web y científicas
Propietario	Microsoft	Oracle	Oracle , (MySql AB)	Postgres Project



Popularidad DBMS- Junio 2024

5.11 ELEMENTOS DE UN DBMS

Entre las principales características de un DBMS que debemos tener en cuenta para seleccionarlo tenemos:

Modelo de datos: Evaluar si el DBMS soporta el modelo de datos requerido por la aplicación, como el modelo relacional, NoSQL.

Escalabilidad: Determinar si el DBMS puede manejar el crecimiento futuro de los datos y el aumento de la carga de trabajo sin comprometer el rendimiento

Rendimiento: Evaluar el rendimiento del DBMS en términos de tiempos de respuesta, eficiencia de consultas, y capacidad de manejar cargas de trabajo intensivas.

Disponibilidad y **confiabilidad**: Considerar las características de alta disponibilidad, como la replicación, el particionamiento y la tolerancia a fallas.

Seguridad: Evaluar las medidas de seguridad, como el control de acceso, la encriptación de datos y la auditoría.

Soporte para consultas complejas: Determinar si proporciona un lenguaje de consulta potente y flexible con operaciones analíticas y agregaciones.

Integridad de datos: que pueda mantener la integridad con implementación de restricciones y reglas de integridad.

Facilidad de uso y administración: Analizar curva de aprendizaje, las herramientas de administración, la documentación y el soporte técnico disponible.

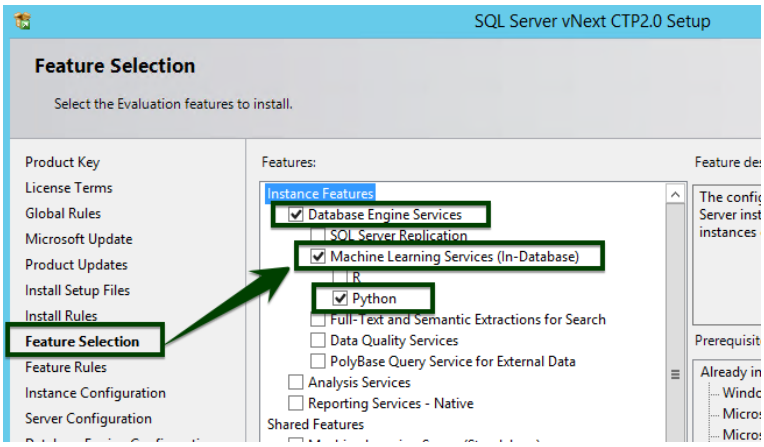
Ecosistema y comunidad: evaluar la comunidad de usuarios y desarrolladores del DBMS, así como la disponibilidad de herramientas adicionales.

Costo y licenciamiento: Evaluar los costos de adquisición, implementación y mantenimiento además del licenciamiento (open source, comercial, suscripción).

Algunos Recursos SQL

Descarga SQL Server- (Recomendado Express)	https://www.microsoft.com/es-es/sql-server/sql-server-downloads
Guía instalación SQL Server	https://sqlserverdb.com/instalar-sql-server/
Descarga MySQL	https://dev.mysql.com/downloads/mysql/
Guia de SQL con MySql	https://www.guru99.com/es/sql.html

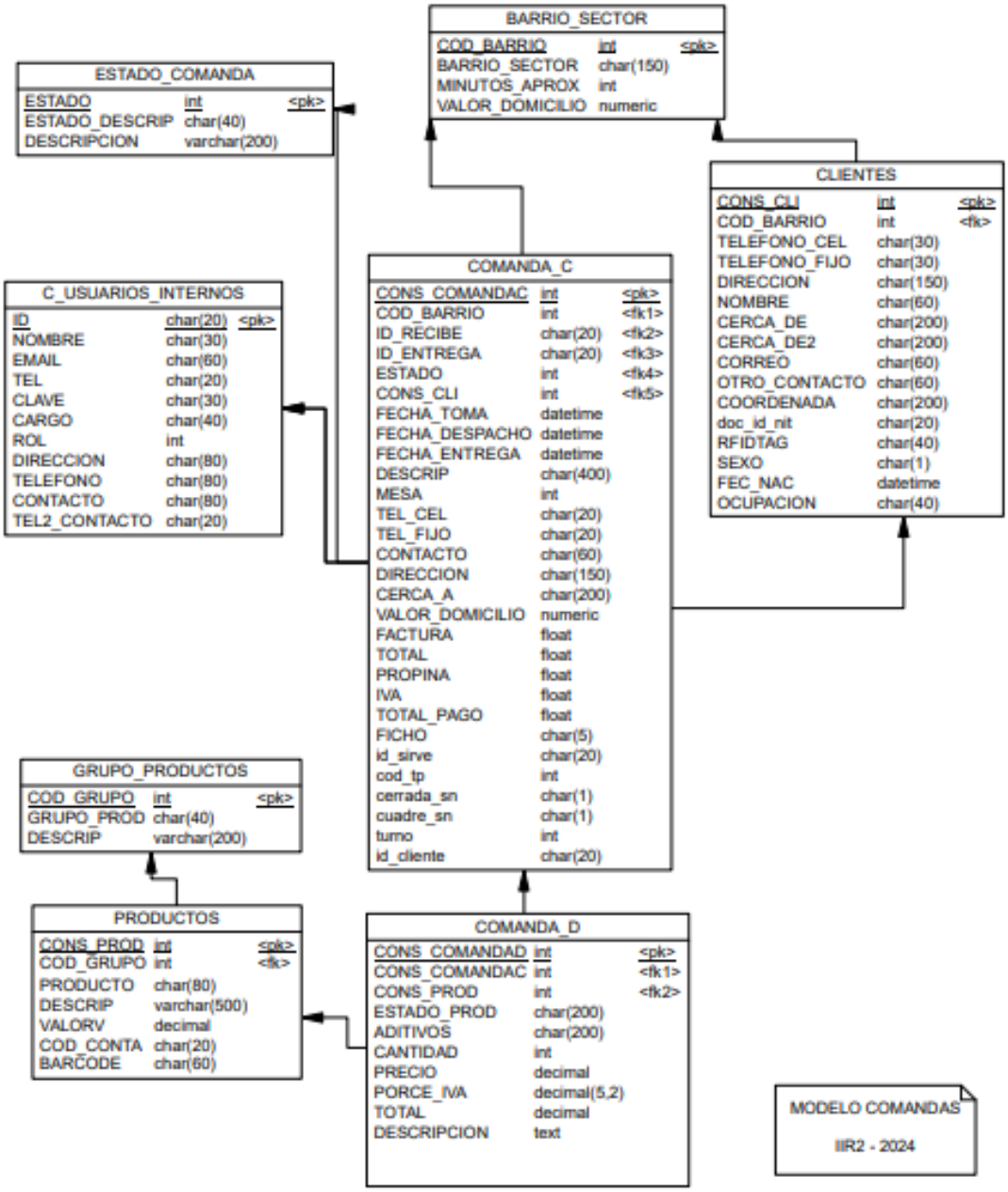
En la instalación de SQL Server puede seleccionar opciones adicionales:



Ejercicio: Modelo “COMANDAS”

Identificar los elementos del modelo para generar consultas SQL sobre este.

SQL



Insertar un nuevo cliente en la tabla CLIENTES

```
INSERT INTO dbo.CLIENTES (NOMBRE, DIRECCION, CORREO, TELEFONO_CEL)
VALUES ('Juan Perez', 'Calle Falsa 123', 'juan.perez@example.com', '1234567890');
```

Actualizar el número de teléfono fijo de un cliente específico:

```
UPDATE dbo.CLIENTES SET TELEFONO_FIJO = '0987654321' WHERE CONS_CLI = 10;
```

Consultar los datos de contacto (nombre, dirección, correo y teléfonos) de todos los clientes

```
SELECT NOMBRE, DIRECCION, CORREO, TELEFONO_CEL, TELEFONO_FIJO FROM
dbo.CLIENTES;
```

En un barrio específico:

```
SELECT NOMBRE, DIRECCION, CORREO, TELEFONO_CEL, TELEFONO_FIJO FROM
dbo.CLIENTES WHERE COD_BARRIO = 5;
```

Eliminar el registro del cliente con el identificador único (CONS_CLI) igual a 15.

```
DELETE FROM dbo.CLIENTES WHERE CONS_CLI = 15;
```

Listar los productos con el grupo de producto al que pertenecen

```
SELECT productos.producto, grupo_prodcutos.grupo_prod
FROM productos, grupo_prodcutos
WHERE productos.cod_grupo = grupo_prodcutos..cod_grupo;
```

Referencias

- https://www.guru99.com/es/sql.html
- https://aprenderbigdata.com/bases-de-datos-relacionales/
- https://www.youtube.com/watch?v=knVwokXITGI
- https://learnsql.es/blog/como-aprender-a-hacer-joins-en-sql/
- https://www.sqlshack.com/es/introduccion-y-resumen-de-la-clausula-de-sql-join/

Herramienta de diseño

- https://online.visual-paradigm.com/es/diagrams/features/erd-tool/
- https://www.drawio.com/

Fecha Creación	Enero 25 2024
Responsable	Plinio Neira Vargas
Revisado por	Sonia Escobar
Fecha Revisión	Febrero 10 2024