BASE DE DATOS

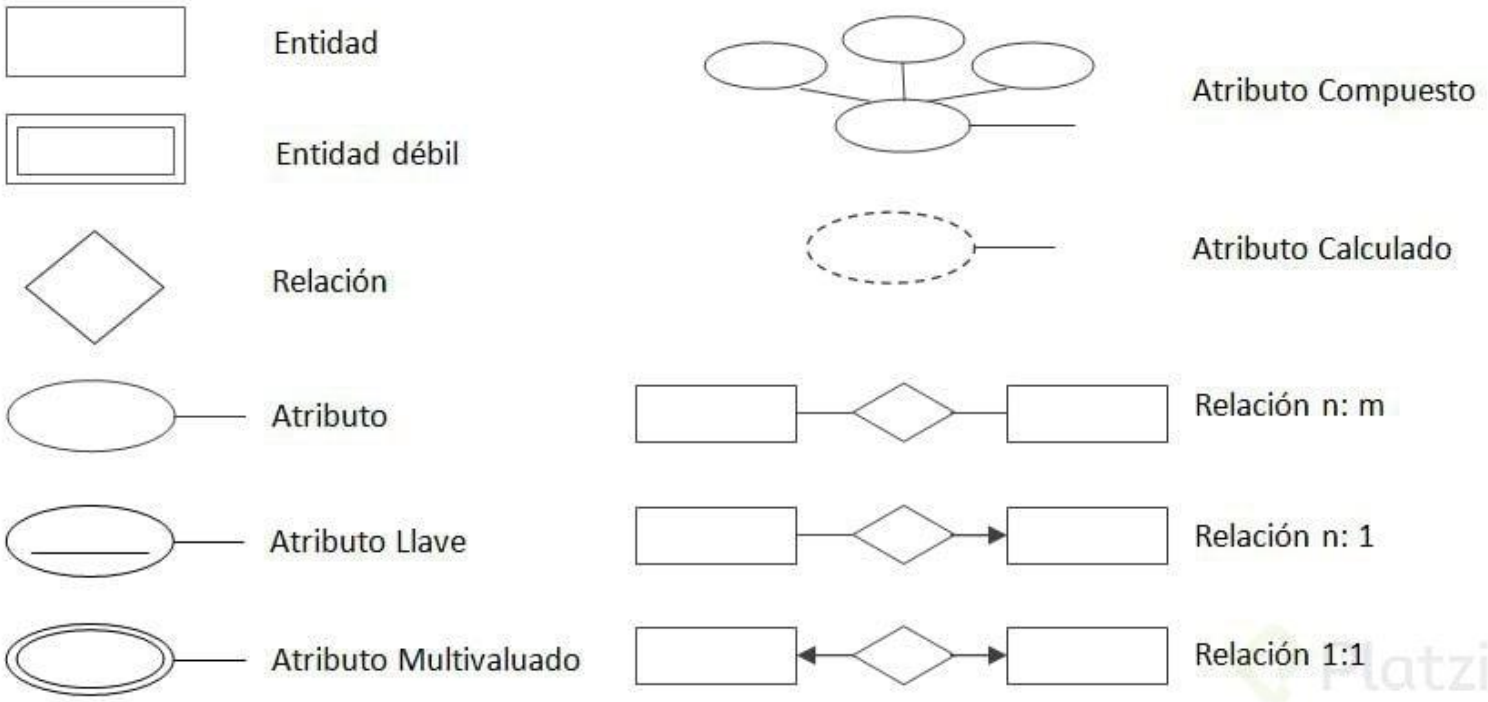
Unidad 4 – 2da parte

Programación avanzada

BASE DE DATOS

Introducción. El modelo Entidad-Relación. Modelo Relacional. Diseño de Base de Datos. Introducción a SQL. Lenguaje de Definición de Datos (DDL). Lenguaje de Manipulación de Datos (DML). Formulación de Consultas con SQL

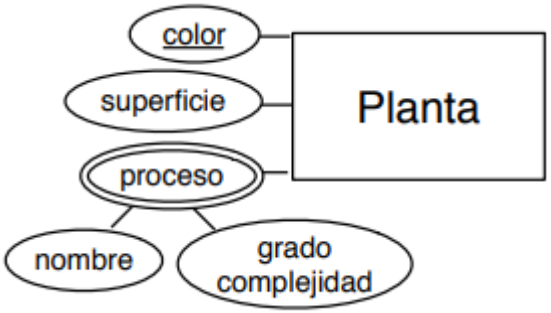
DIAGRAMA ENTIDAD-RELACIÓN



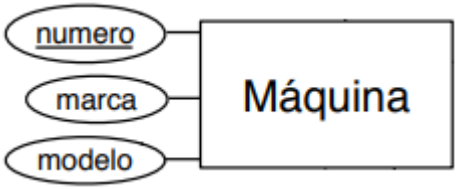
EJERCICIO. FÁBRICA DE PELOTAS

- Solicitan nuestros servicios para resolver el almacenamiento de datos de un sistema de gestión de la producción de una fábrica de pelotas.
- La fábrica se compone de una serie de plantas, cada una identificada por un color. De las plantas conocemos la superficie en metros cuadrados y la lista de procesos que se llevan a cabo dentro de ellas; de estos procesos sólo conocemos su nombre y un grado de complejidad asociado.
- Dentro de cada planta se encuentran las máquinas. Cada máquina es de una marca y un modelo, y se identifica por un número; este número es único a lo largo de todas las plantas.
- Cada máquina es operada por técnicos, debemos conocer en qué rango de fechas los técnicos estuvieron asignados a esa máquina, y además en qué turno (mañana, tarde o noche).
- De los técnicos conocemos su DNI, nombre, apellido y fecha de nacimiento, aparte de una serie de números telefónicos de contacto.
- Existen situaciones normales en las que una máquina sale de servicio y debe ser reparada, lo único que nos interesa conocer aquí es cuál otra máquina está asignada para tomar el trabajo que ella no puede realizar.

DIAGRAMA E-R: ENTIDADES



*La fábrica se compone de una serie de **plantas**, cada una identificada por un color. De las plantas conocemos la superficie en metros cuadrados y la lista de procesos que se llevan a cabo dentro de ellas; de estos procesos sólo conocemos su nombre y un grado de complejidad asociado.*



*Cada **máquina** es de una marca y un modelo, y se identifica por un número; este número es único a lo largo de todas las plantas*

*De los **técnicos** conocemos su DNI, nombre, apellido y fecha de nacimiento, aparte de una serie de números telefónicos de contacto.*

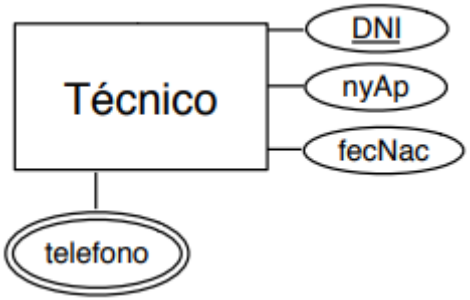
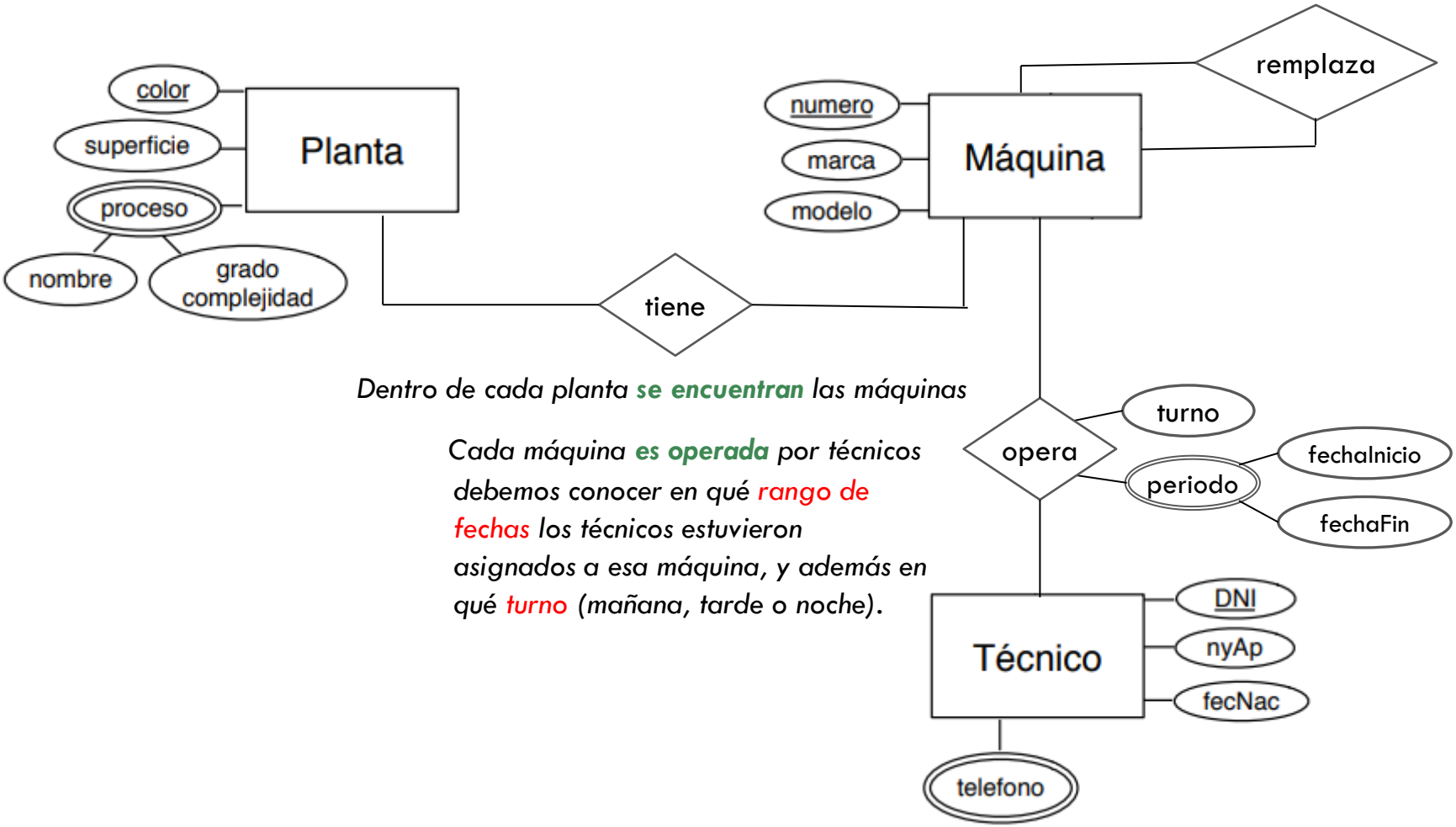


DIAGRAMA E-R: RELACIONES

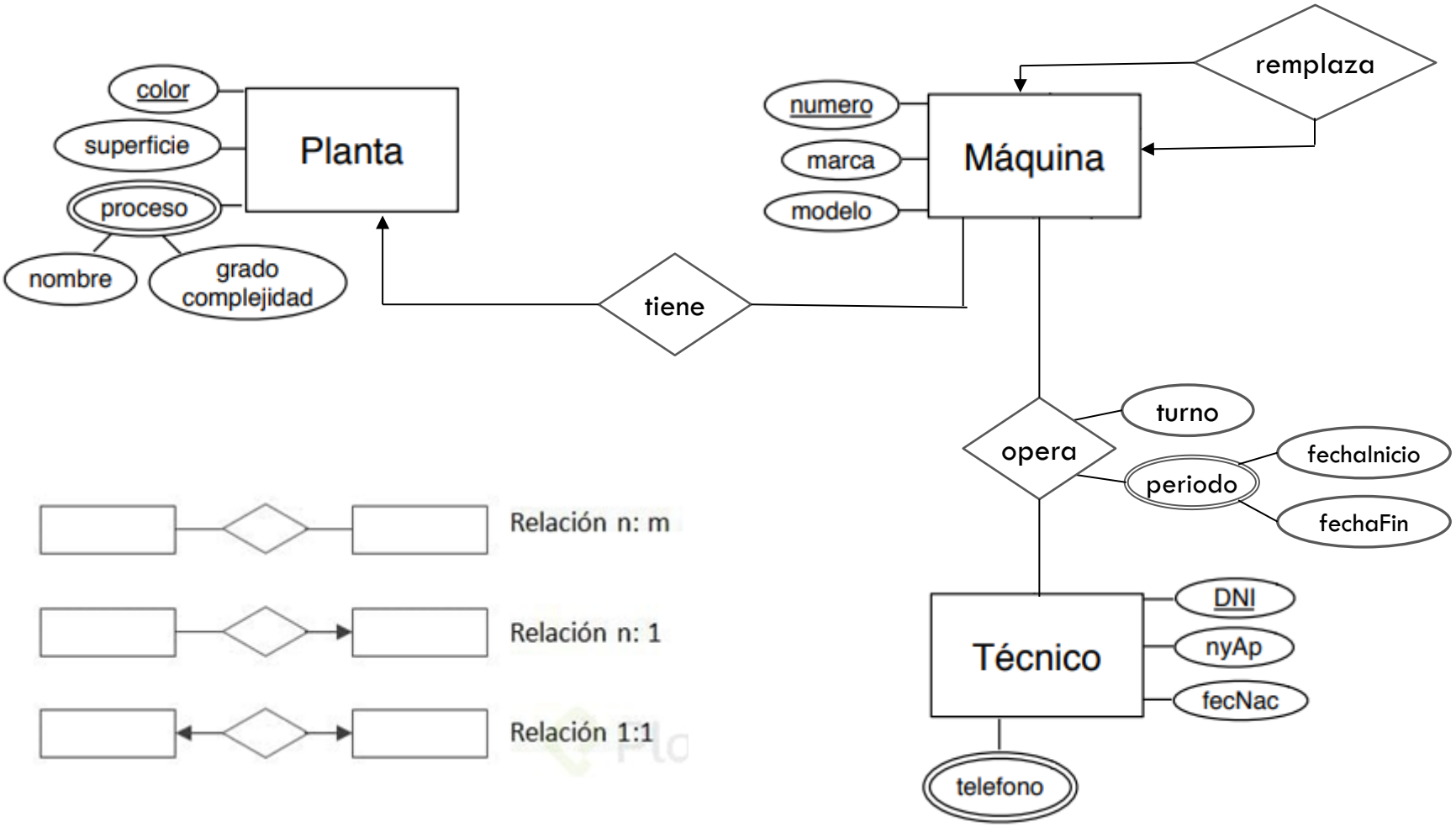
Existen situaciones normales en las que una máquina sale de servicio y debe ser reparada, lo único que nos interesa conocer aquí es cuál otra máquina está asignada ***para tomar el trabajo*** *que ella no puede realizar*



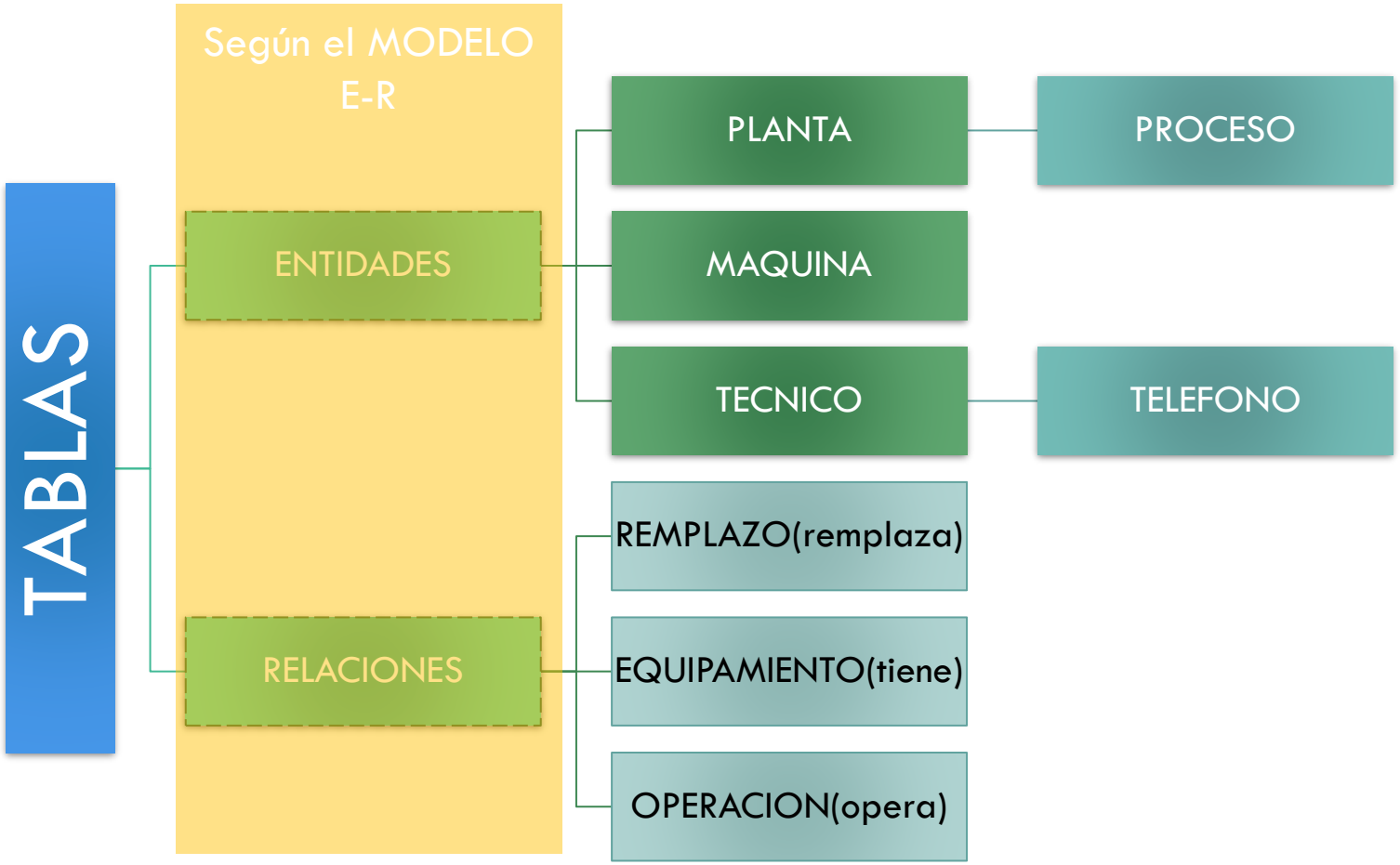
Dentro de cada planta ***se encuentran*** *las máquinas*

Cada máquina ***es operada*** *por técnicos debemos conocer en qué rango de fechas los técnicos estuvieron asignados a esa máquina, y además en qué turno (mañana, tarde o noche).*

DIAGRAMA E-R: #RELACIONES

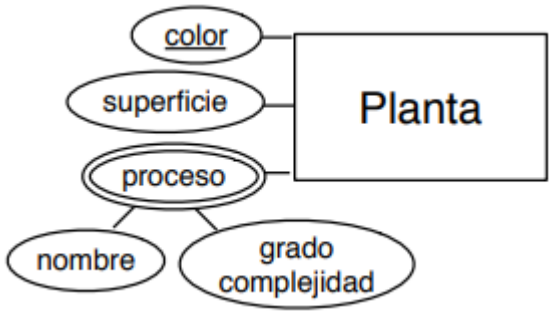


MODELO RELACIONAL



MODELO RELACIONAL. TABLAS

ENTIDAD: Planta



Como ***proceso*** *es un* ***atributo multivaluado***

TABLA: Planta

CAMPOS	TIPO DE DATO
Color	Cadena
Superficie	Numero Real

TABLA: Proceso

CAMPOS	TIPO DE DATO
Color	Cadena
Nombre	Cadena
GradoComplejidad	Cadena

MODELO RELACIONAL. TABLAS

ENTIDAD: Máquina

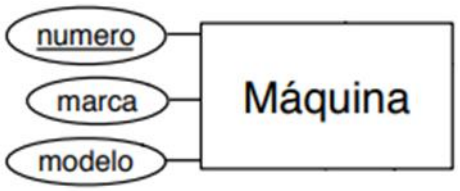


TABLA: Máquina

CAMPOS	TIPO DE DATO
Numero	Entero
Marca	Cadena
Modelo	Cadena

MODELO RELACIONAL: TABLAS

ENTIDAD: Técnico

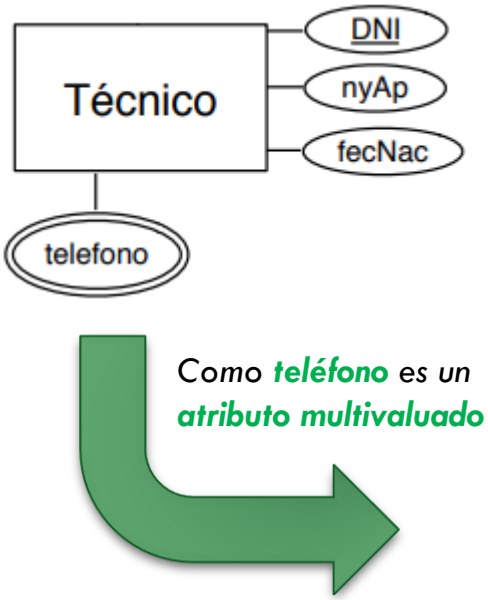


TABLA: Técnico

CAMPO	TIPO DE DATO
DNI	Entero
NombreYApellido	Cadena
FecNac	Fecha

TABLA: Teléfono

CAMPOS	TIPO DE DATO
DNI	Entero
TipoTel	Cadena
NroTelefono	Cadena

MODELO RELACIONAL. TABLAS

RELACION: tiene

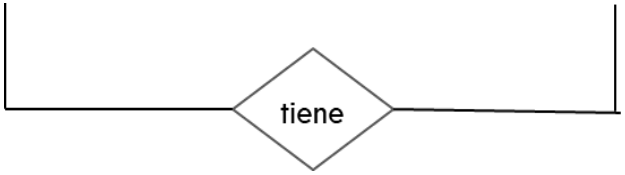


TABLA: EQUIPAMIENTO

CAMPOS	TIPO DE DATO
ColorPlanta	Cadena
NumeroMaquina	Entero

MODELO RELACIONAL. TABLAS

RELACION: opera

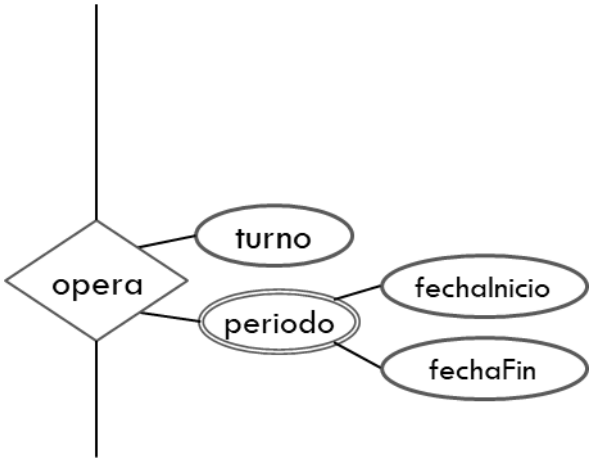


TABLA: OPERACION

CAMPOS	TIPO DE DATO
DNITecnico	Entero
NumeroMaquina	Entero
Turno	Cadena
FechaInicio	Fecha
FechaFin	Fecha

MODELO RELACIONAL. TABLAS

RELACION: remplaza

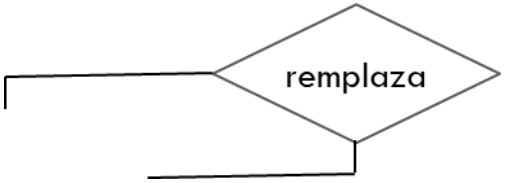


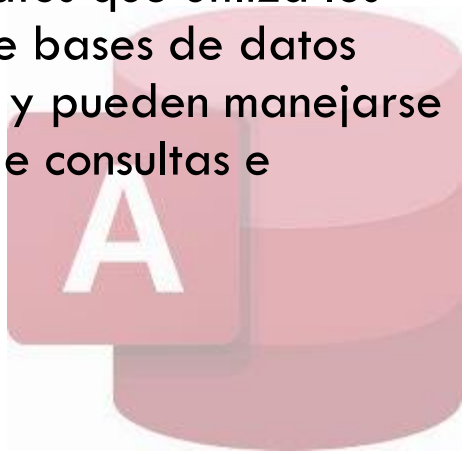
TABLA: REMPLAZO

CAMPOS	TIPO DE DATO
NumeroMaquina1	Entero
NumeroMaquina2	Entero

DBMS

ACCESS

Microsoft Access es un sistema de gestión de bases de datos incluido en las ediciones profesionales de la suite Microsoft Office. Access es un gestor de datos que utiliza los conceptos de bases de datos relacionales y pueden manejarse por medio de consultas e informes.



MySQL

MySQL es un sistema de administración de bases de datos relacionales. Es un software de código abierto desarrollado por Oracle. Se considera como la base de datos de código abierto más utilizada en el mundo.

SQL: *LENGUAJE DE CONSULTA ESTRUCTURADO*

SQL (Structured Query Language) es un lenguaje estándar e interactivo de acceso a bases de datos relacionales, permite realizar distintas operaciones en ellas, gracias a la utilización del álgebra y del cálculo relacional.

SQL nos ofrece la posibilidad de realizar ***consultas*** con el fin de recuperar información de las bases de datos, realizando este proceso de forma sencilla.

Las consultas permiten ***seleccionar, insertar, actualizar, averiguar la ubicación de los datos, …***

SQL (STRUCTURED QUERY LANGUAGE)

SQL es un ***lenguaje de consulta***, un lenguaje en el que el usuario solicita información de la base de datos, suelen ser de un nivel superior a los lenguajes de programación.

Además de consultas, con SQL, es posible definir la estructura de los datos, modificar los datos de la base de datos y especificar restricciones de seguridad.

SQL: CONSIDERACIONES

En SQL no se distingue entre mayúsculas y minúsculas. El final de una instrucción o sentencia lo marca el signo de punto y coma.

Las sentencias SQL (SELECT, INSERT, …) se pueden escribir en varias líneas siempre que las palabras no sean partidas.

Los comentarios en el código SQL pueden ser de 2 tipos:

```
/*
  Esto es un comentario
  de varias líneas.
   Fin.
*/
 -- Esto es un comentario de una línea
```

SQL: TIPOS DE DATOS EN SQL (DOMINIOS)

char(n). Cadena de caracteres de longitud fija n, especificada por el usuario

varchar(n). Cadena de caracteres de longitud variable con una longitud máxima n especificada por el usuario.

int. Integer, un subconjunto finito de los enteros depende de la máquina.

smallint. Small integer (un subconjunto dependiente de la máquina del tipo dominio entero).

real, double precision. Número en coma flotante y números en coma flotante de doble precisión, con precisión dependiente de la máquina.

numeric(p,d). Un número en coma fija, cuya precisión la especifica el usuario. El número está formado por p dígitos (más el signo) y de esos p dígitos, d pertenecen a la parte decimal.

float(n). Un número en coma flotante cuya precisión es de al menos n dígitos

date: Fechas, contiene un año (4 dígitos), mes y día

time: Hora del día, en horas, minutos y segundos.

timestamp: fecha y hora del día

interval: periodo de tiempo

SQL (STRUCTURED QUERY LANGUAGE)

Lenguaje de definición de datos (DDL): proporciona órdenes para definir, modificar o eliminar los distintos objetos de la base de datos (tablas, vistas, índices…). ***(CREATE, DROP, ALTER, ..)***

SQL-DDL: CREATE TABLE

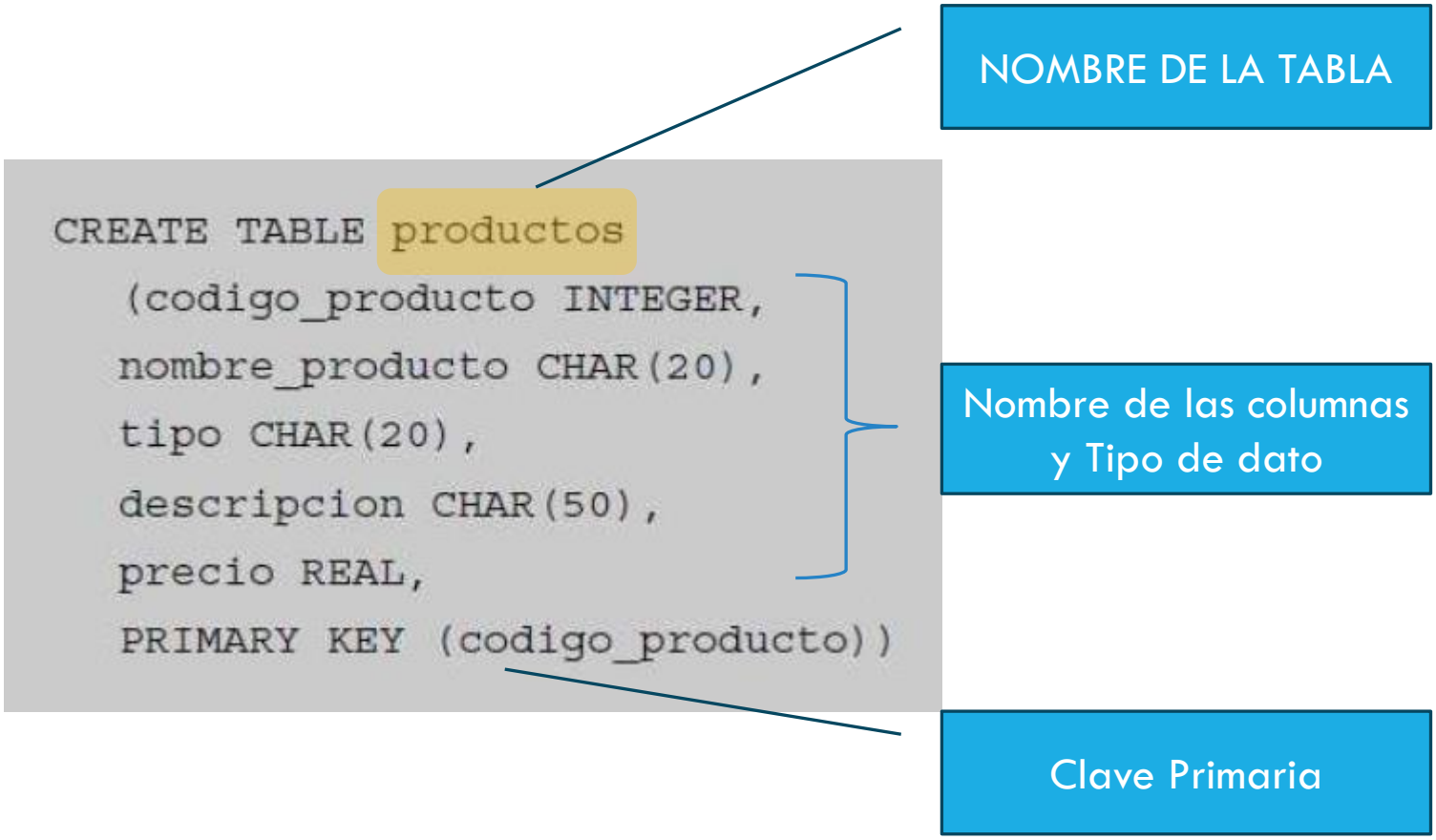
Este comando se lo utiliza para crear las tablas bases que forman la esencia de una base de datos relacional. Consiste en uno o más encabezados (headings) de columna, que proporcionan el nombre y el tipo de datos de la columna, y cero o más filas de datos, contienen un valor de datos del tipo de datos especificados para cada columna.

La forma de dicho comando es:

CREATE TABLE nombre_tabla (nombre_columna tipo_columna, [restricciones – **NULL/NOT NULL, DEFAULT, UNIQUE, CHECKJ**],……,

[restricciones tabla – **PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK….**])

SQL-DDL: CREATE TABLE



SQL-DDL: CREATE INDEX

Los índices nos ayudan a obtener datos de las tablas en forma más rápida. Al dirigirnos primero al índice ***ahorramos tiempo*** logrando así un método más eficiente para ubicar la información que necesitamos.

Sin un índice, el sistema de base de datos lee a través de toda la tabla (este proceso se denomina "escaneo de tabla") para localizar la información deseada. Con el índice correcto en su lugar, el sistema de base de datos puede entonces primero dirigirse al índice para encontrar de donde obtener los datos, y luego dirigirse a dichas ubicaciones para obtener los datos necesarios. Esto es mucho más rápido.

Por lo tanto, generalmente se recomienda crear índices en tablas. Un índice puede cubrir una o más columnas. La sintaxis general para la creación de un índice es:

CREATE INDEX nombre_indice **ON** nombre_tabla (nombre_columna);

SQL-DDL: CREATE INDEX

Si se quiere crear un índice con respecto al Apellido, la sintaxis quedaría de la siguiente manera:

CREATE INDEX idxapellido **ON** CLIENTE(Apellido);

Si se quiere que el índice se dé mas de una columna se escribe:

CREATE INDEX idxdireccion **ON** CLIENTE(Ciudad,Pais);

CREATE INDEX idxnombrecompleto **ON** CLIENTE(Nombre,Apellido);

Nombre de Columna	Tipo de Datos
Nombre	Char(50)
Apellido	Cha(50)
Direccion	Char(50)
Ciudad	Char(50)
Pais	Char(50)
FechaNacimiento	datetime

SQL-DDL: ALTER TABLE

Una vez creada una tabla, los usuarios pueden encontrarla más útil si contiene un ítem de datos adicional, no tiene una columna particular o tiene diferentes restricciones. Aquí, la naturaleza dinámica de una estructura de base de datos relacional hace posible cambiar las tablas base existentes.

Por ejemplo, para agregar una nueva columna a la derecha de la tabla se usa el comando de la siguiente forma:

ALTER TABLE nombre_tabla **ADD** nombre_columna tipo_columna;

Ejemplo: se decide agregar a la tabla Productos la columna Estado, cuyo tipo de datos es char(15)

ALTER TABLE productos **ADD** Estado **CHAR**(15);

Por lo que la tabla quedaría de la siguiente forma:

PRODUCTOS (codigo_producto, nombre_producto, tipo, descripcion, precio, Estado)

SQL-DDL: DROP TABLE

Las tablas se pueden eliminar en cualquier momento mediante el comando:

DROP TABLE nombre_tabla;

Cuando se ejecuta este comando se remueven la tabla en si y todos los registros contenidos en ella.

Además, todos los índices y todas las vistas que dependen de ella se deben de eliminar.

También se puede eliminar cualquier índice con la siguiente secuencia:

DROP INDEX nombre_indice;

SQL-DDL: TIPOS DE DATOS

Los tipos de datos disponibles concluyen varios tipos numéricos, cadenas de caracteres de longitud fija y de longitud variable, cadenas de bits y tipos definidos por el usuario. Los tipos de datos disponibles varían de ***DBMS*** a ***DBMS***.

ORACLE	DB2	Microsoft SQL Server	Microsoft Access
CHAR(N), VARCHAR2(N), NUMBER(N,D), DATE BLOB (gran objeto binario).	SMALLINT, INTEGER, BIGINT, DECIMAL/NUMERIC, REAL, DOUBLE, CHAR(N), VARCHAR(N), LONG VARCHAR, CLOB, GRAPHIC, DBCLOB, BLOB, DATE, TIME TIMESTAMP.	NUMERIC, BINARY, CHAR, VARCHAR, DATE-TIME, MONEY, IMAGE otros.	NUMBER, TEXT, MEMO, DATE/TIME, CURRENCY, YES/NO otros.

SQL (STRUCTURED QUERY LANGUAGE)

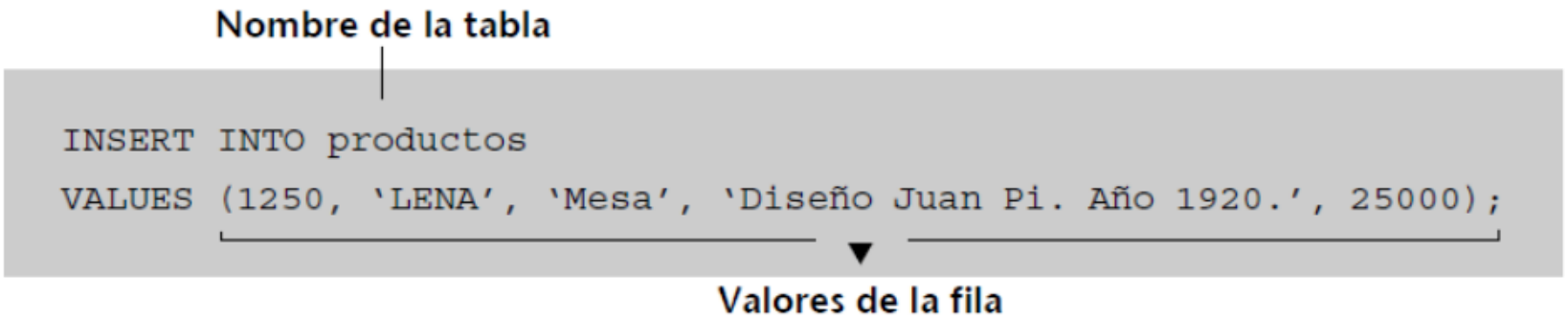
Lenguaje de Manipulación de Datos (DML): proporciona órdenes para insertar, suprimir y modificar registros o filas de las tablas. También contempla la realización de consultas sobre la BD. **(SELECT, INSERT, UPDATE y DELETE)**

SQL-DML: INSERT

Una sentencia INSERT de SQL agrega uno o más registros a una (y solo una) tabla en una base de datos relacional.

Forma Básica

INSERT INTO nombre_tabla (“nombre_columna1”,”[nombre_columna2…..]”) **VALUES** (“valor1”, “[valor2...]”);



SQL-DML: UPDATE

Una sentencia **UPDATE** de SQL es utilizada para **modificar** los valores de un conjunto de registros existentes en una tabla.

Forma Básica

UPDATE nombre_tabla **SET** “nombre_columna1”=“nuevo_valor,” [“nombre_columna2”=“nuevo_valor2,….”] **WHERE** condición;

Ejemplo

UPDATE proveedores **SET** emailpro=mailto:il@herrera.unju.edu.ar **WHERE** codigopro=4000;

SQL-DML: DELETE

La sentencia **DELETE** borra uno o más registros existentes en una tabla.

Forma Básica

DELETE FROM nombre_tabla **WHERE** nombre_columna1=valor1;

Ejemplo

DELETE FROM proveedores **WHERE** codigopro=4000;

SQL-DML: SELECT

La sentencia **SELECT** nos permite consultar los datos almacenados en una tabla de la base de datos.

El formato de la sentencia **SELECT** es:

SELECT [**ALL** | **DISTINCT**] nombre_columna [,nombre_columna…]

FROM nombre_tabla|nombre_vista [nombre_tabla|nombre_vista…]

[**WHERE** condicion [AND|OR condicion]]

[**GROUP BY** nombre_columna [nombre_columna …]]

[**HAVING** condicion [**AND**|**OR** condicion]]

[**ORDER BY** nombre_columna|índice_columna [ASC | DESC],
…[[nombre_columna|índice_columna[ASC |DESC]]]]

SQL-DML: SELECT

SELECT: Palabra clave que indica que la sentencia de SQL que queremos ejecutar es de selección.

ALL: Indica que queremos seleccionar todos los valores. Es el valor por defecto y no suele especificarse casi nunca.

DISTINCT: Indica que queremos seleccionar sólo los valores distintos.

FROM: Indica la tabla (o tablas) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denomina a la consulta "consulta combinada" o "JOIN". En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.

WHERE: Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos AND y OR.

GROUP BY : Especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas.

HAVING: Especifica una condición que debe cumplirse para los datos. Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Su funcionamiento es similar al de WHERE pero aplicado al conjunto de resultados devueltos por la consulta. Debe aplicarse siempre junto a GROUP BY y la condición debe estar referida a los campos contenidos en ella.

ORDER BY :Presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC (orden descendente). El valor predeterminado es ASC.

EJEMPLO

```
SELECT matricula, marca, modelo, color, numero_kilometros, num_plazas FROM tCoches WHERE matricula= 'AA125KO';
```

SQL-DML: SELECT

OPERADORES

- AND, OR, NOT
- %LIKE%
- <, >, =, <>, !=
- IN
- BETWEEN
- IS [NOT] NULL

SQL (STRUCTURED QUERY LANGUAGE)

Lenguaje de Control de Datos (DCL): permite establecer derechos de acceso de los usuarios sobre los distintos objetos de la base de datos. Lo forman las instrucciones **GRANT** y **REVOKE.**

SQL- DCL

REVOKE

- Este comando crea un objeto dentro de la base de datos.

GRANT

- Este comando permite modificar la estructura de un objeto.

DANY

- Este comando elimina un objeto de la base de datos. Se puede combinar con la sentencia ALTER.

Fin Unidad 4