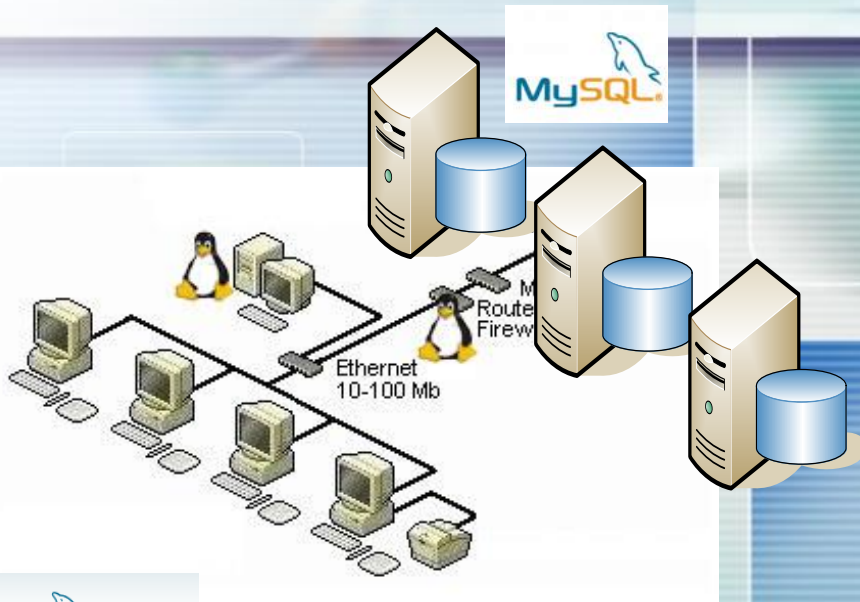




# Universidad Privada de Tacna

## Escuela Profesional de Ingeniería de Sistemas

### Base de Datos



MySQL  
Administrator

MySQL  
Query Browser

IV Ciclo

Ing. Enrique Lanchipa Valencia

# Resumen



¿Qué es una Base de Datos?



SGBD



Componentes de una BD



Lenguaje SQL



Mysql



Ventajas - Desventajas

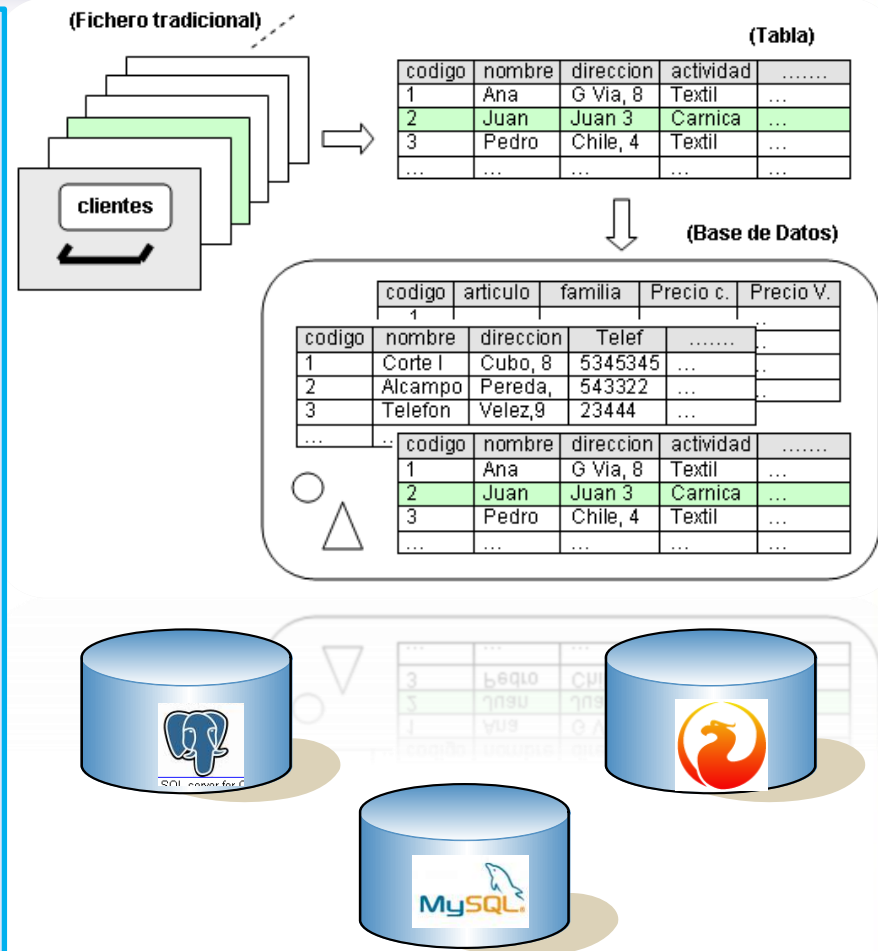


Características.



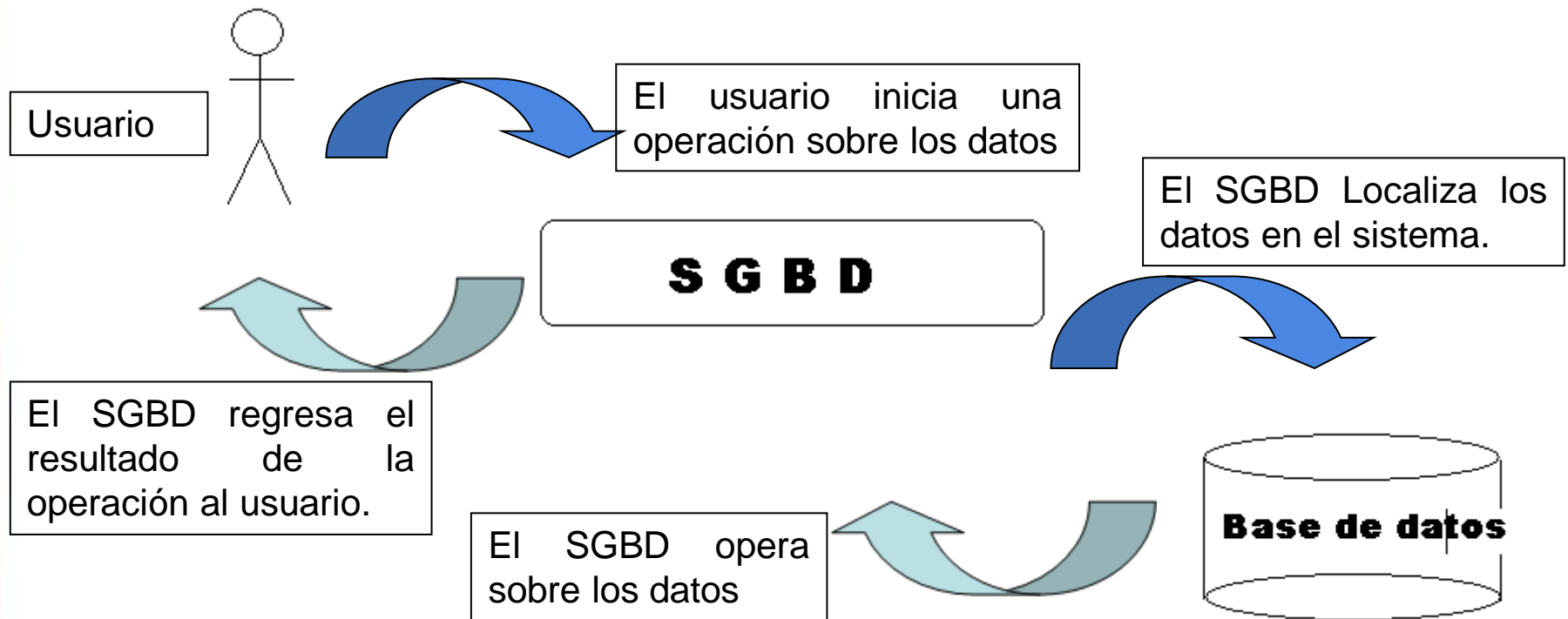
# ¿Qué es una Base de Datos?

- ▶ Conjunto de datos de la empresa memorizado en un ordenador, que es utilizado por numerosas personas y cuya organización está regida por un modelo de datos (**Flory, 1982**).
- ▶ De manera mas técnica, una base de datos es un conjunto de varios tipos de datos organizados e interrelacionados (**Elmars, R, Navathe S.B.1989**).
- ▶ A manera de teoría de conjuntos, es un conjunto de datos de diferentes ámbitos, organizados sistemáticamente, es decir, siguen ciertas reglas.
- ▶ Colección no redundante de datos que son compartidos por diferentes programas de aplicación (**Howe, 1983**).



# Definición de Manejador de Bases de Datos

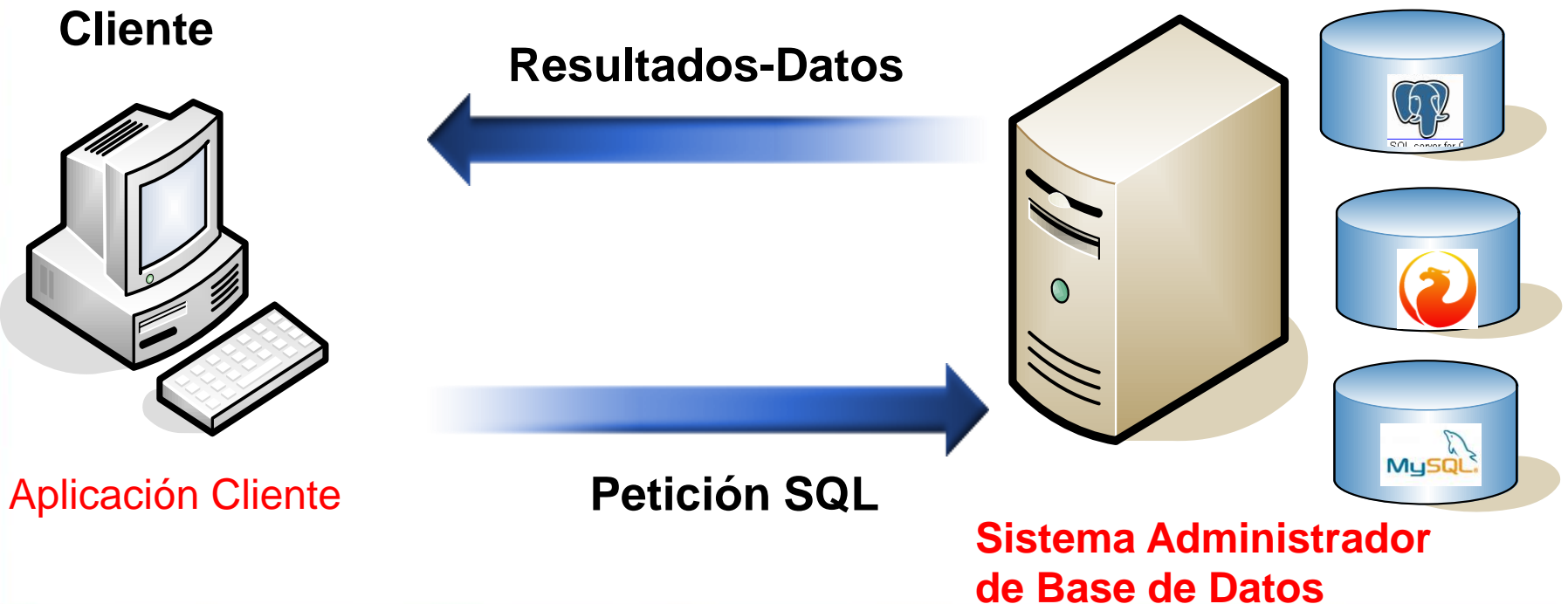
Un **manejador de base de datos**, conocido con las siglas **SGBD** – Sistema Gestor de Base de Datos o en ingles **DBMS** – Database Manager System, **es un software que actúa como interfaz entre los datos almacenados en forma binaria en una base de datos y el usuario que desea manejar tales datos.**



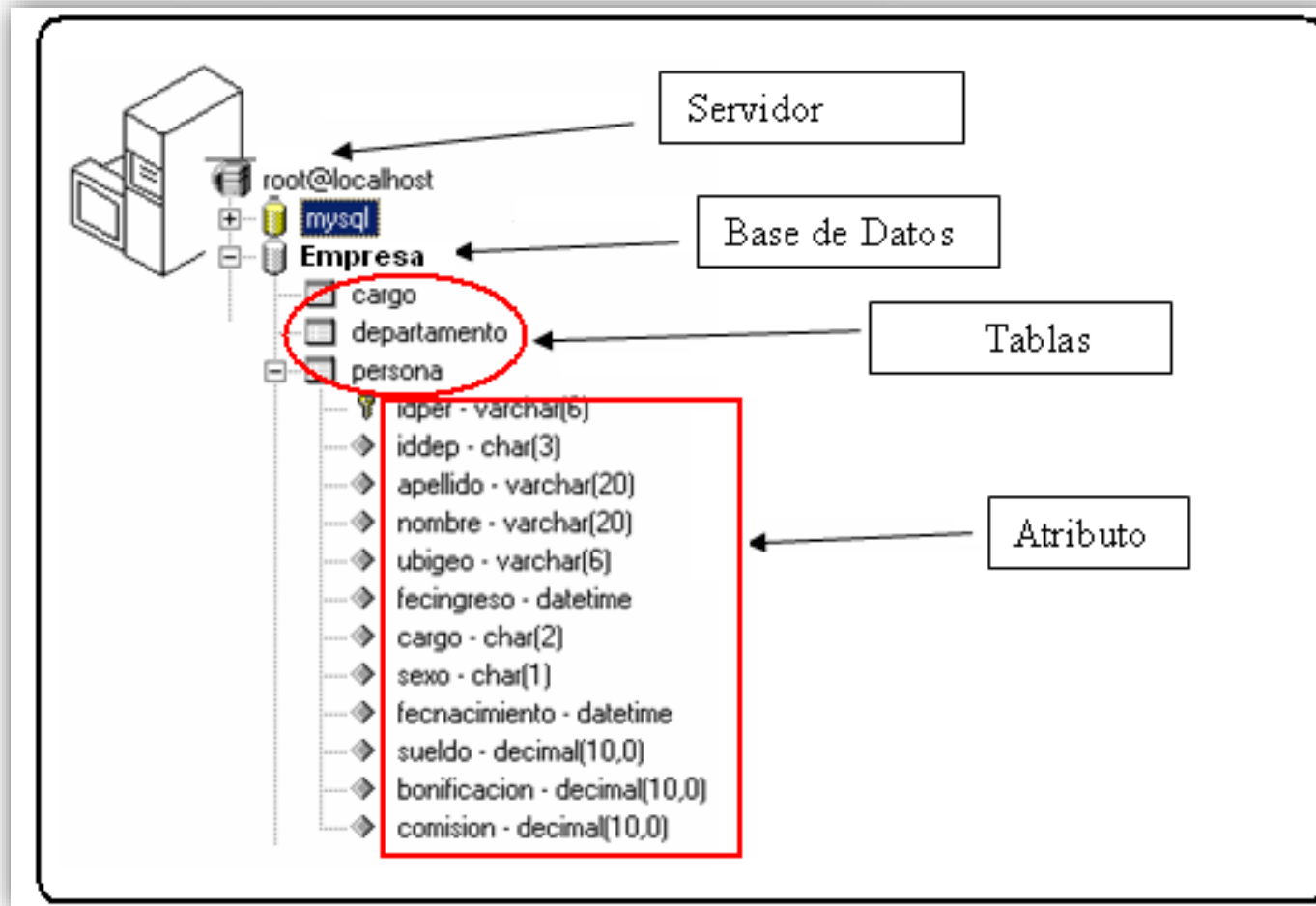


# Definición de Manejador de Bases de Datos

Los **Sistemas Gestores de Bases de Datos** son un tipo de software muy específico, dedicado a servir de interfaz entre la Base de datos y el usuario, las aplicaciones que la utilizan. Se compone de un lenguaje SQL.



# Componentes de una Base de Datos

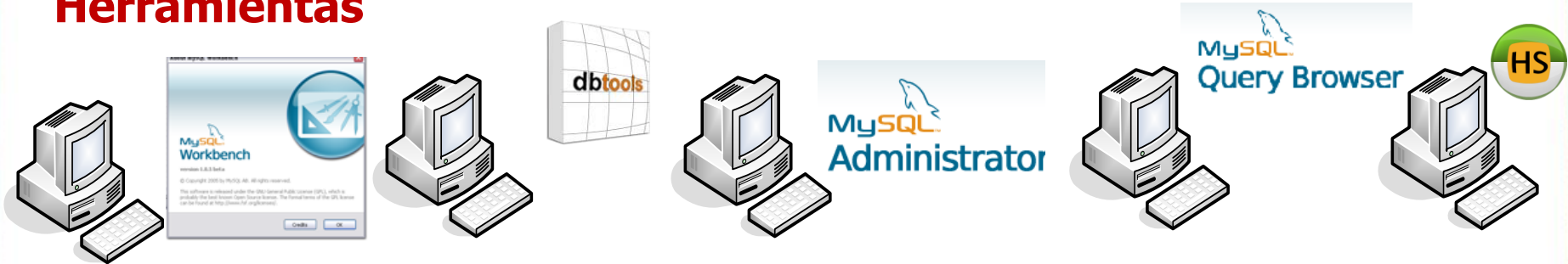


# Bases de Datos Comerciales

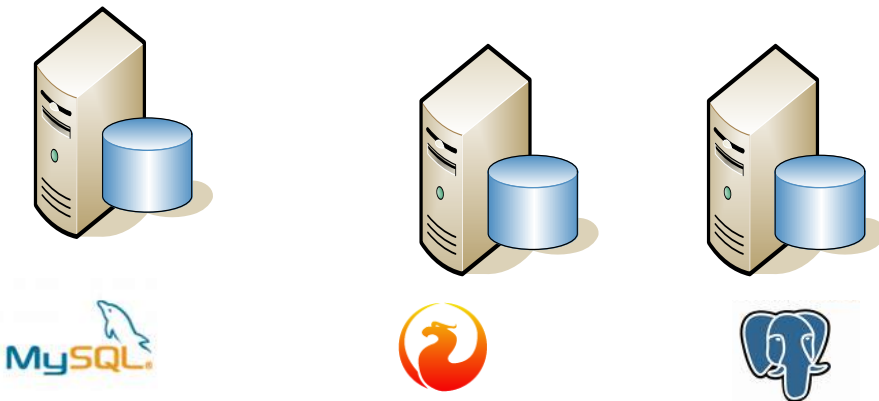


# Bases de Datos Libres

## Herramientas



## Servidores





# SQL – Lenguaje de Base de Datos

**SQL.** Es el lenguaje estándar para el manejo de base de datos. **SQL - Structured Query Language.**

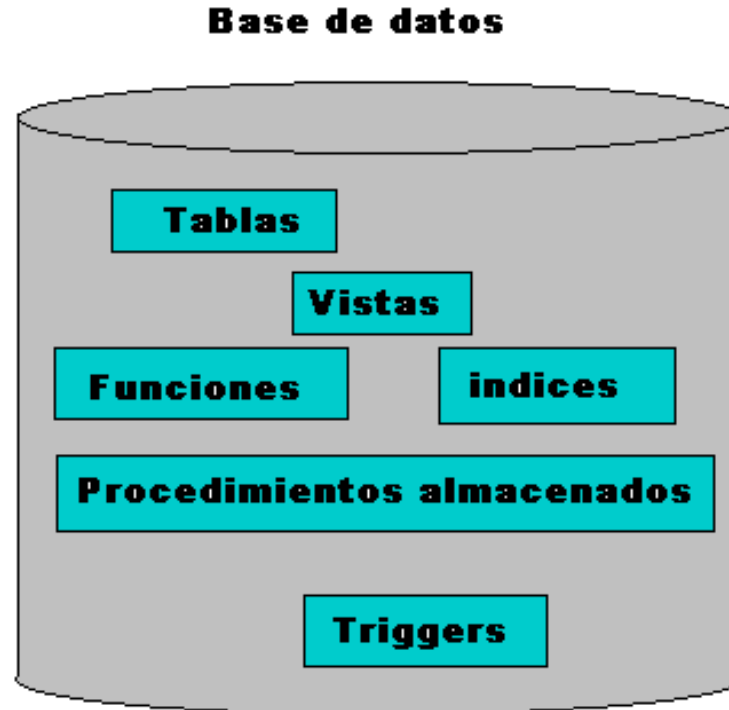
SQL se divide en dos Sub-lenguajes.

**DDL (Data Definition Language).** Lenguaje para la definición de objetos de la base de datos. Agrupa a las operaciones **CREATE DATABASE, CREATE TABLE, CREATE INDEX** ETC...

**DML (Data Manipulation Language).** Lenguaje para la manipulación de datos agrupa a las operaciones **SELECT, INSERT, UPDATE y DELETE.**

## Objetos de una Base de Datos

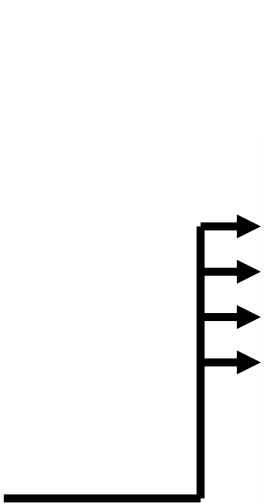
Las bases de datos están compuestas básicamente por 6 objetos: **Tablas, Vistas, Funciones, Índices, Procedimientos almacenados y Trigger**



## Objeto: Tabla

Las **tablas** son los objetos principales de una base de datos, pues **son la estructura Física donde se almacenan los datos**. Las tablas contienen **registros** los cuales contienen **campos**.

**Columnas = Campos**



ClaveProducto	NombreProducto	PrecioUnitario
2021	Producto 1	10.22
2022	Producto 2	100.25
2023	Producto 3	10.50
2024	Producto 4	11.00
2025	Producto 5	15.00
2026	Producto 6	89.45
2027	Producto 7	80.00
2028	Producto 8	12.00
2029	Producto 9	25.00

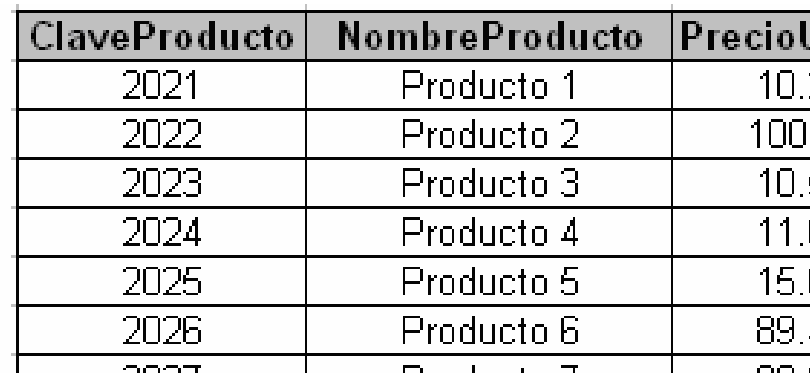
**Filas = Registros**

**TABLA**

## Objeto: Vistas

Las **vistas** son tablas derivadas de otras tablas (básicas o virtuales), Las vistas contienen **registros** los cuales contienen **campos**.

**Columnas = Campos**



ClaveProducto	NombreProducto	PrecioUnitario
2021	Producto 1	10.22
2022	Producto 2	100.25
2023	Producto 3	10.50
2024	Producto 4	11.00
2025	Producto 5	15.00
2026	Producto 6	89.45
2027	Producto 7	80.00
2028	Producto 8	12.00
2029	Producto 9	25.00

**Filas = Registros**

**VISTAS**

# Definiendo Objetos de una base de datos (DDL)



# Introducción a SQL

## Sintaxis

```
CREATE DATABASE NOMBRE_BASE
```

**Ejemplos:** crear una base llamada “BasePrueba”:

```
CREATE DATABASE BasePrueba
```

Crear una base de datos llamada “Base\_Tres”

```
CREATE DATABASE BaseTres
```

## Importante.

Los nombres de las base de datos y así como también los nombre de las tablas de la base de datos no deben contener espacios en blanco.

# Introducción a SQL

Sentencia **CREATE TABLE**. Construye una tabla

## Sintaxis

```
CREATE TABLE nombreDeTabla (  
                                Campo1 tipo de dato ,  
                                Campo2 Tipo de dato,  
                                Campo2 Tipo de dato,...  
                                )
```

## Ejemplo:

```
CREATE TABLE TABLA1 (  
                                Campo1 int,  
                                Campo2 int,  
                                Campo3 char(3)  
                                )
```

```
CREATE TABLE Proveedores (  
                                ClaveProveedor int,  
                                Nombre char(40)  
                                )
```

# Modelado

Al hacer el análisis se elabora un diagrama que describa la tabla.

**Figura para modelar una tabla.**  
**Modelo Lógico**

**Nombre\_tabla**

<b>Campo_1</b>
<b>campo_2</b>
<b>Campo_3</b>
.
.



**Figura para modelar una tabla.**  
**Modelo Físico**

**Nombre\_tabla**

<b>Campo_1</b> TipoDato
<b>Campo_1</b> TipoDato
<b>Campo_1</b> TipoDato
.
.
.

# Modelado

## Tabla Alumnos Modelo Lógico

Alumnos

Matricula

Nombre

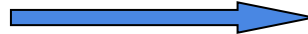
Apellidos

Fecha\_Nac

Telefono

Direccion

Pasamos del  
modelo lógico al  
modelo Físico



## Tabla Alumnos Modelo Físico

Alumnos

Matricula: int

Nombre: char(20)

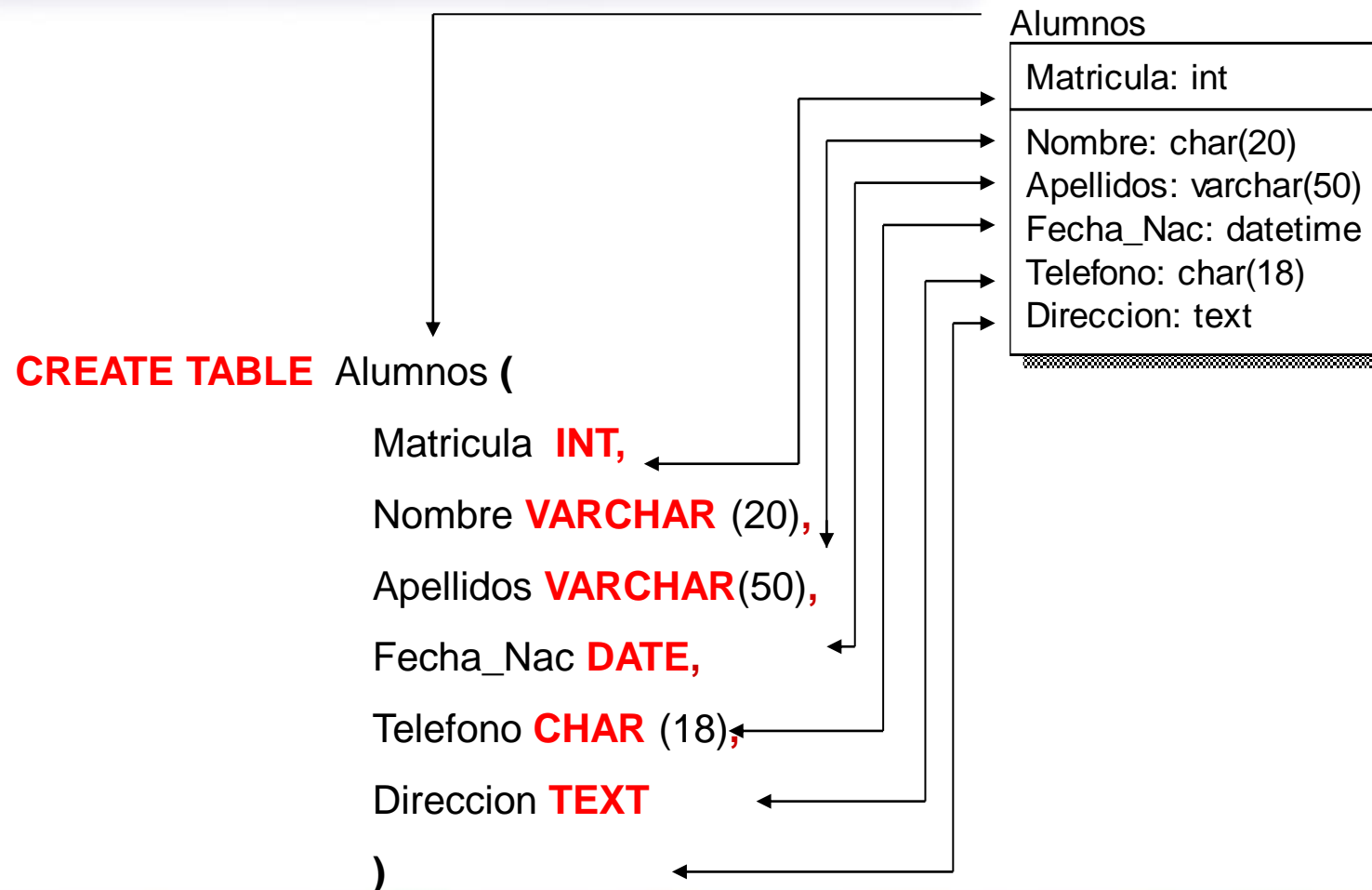
Apellidos: varchar(50)

Fecha\_Nac: datetime

Telefono: char(18)

Direccion: text

# Implementación del Modelo





# **Tipos de datos para los campos de una Tabla**

## Tipos de datos. CADENA DE CARACTERES

**CHAR.** Cadena de caracteres de **longitud fija**.

### Sintaxis.

Nombre\_Campo **CHAR** ( Numero de caracteres )

### Ejemplos.

ClaveEmp **CHAR**(4) => '55JR', 'FFF1', '0001'

Telefono **CHAR**(10) => '5510174536', '5556581213'

ClaveEmp **CHAR**(4) => '5JR', '00F', '01' - - **datos aceptados**

ClaveEmp **CHAR**(4) => '550JR' - - **El dato es truncado a '550J'**  
- - **pero es insertado.**

## Tipos de datos. CADENA DE CARACTERES

Nom **CHAR** (10)



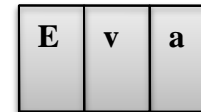
Nom **VARCHAR** (10)



Nom **CHAR** (10) => **'Eva'**



Nom **VARCHAR** (10) => **'Eva'**



### Nota:

El limite, para **CHAR** Y **VARCHAR** es de 8 000 caracteres

## Tipos de datos. CADENA DE CARACTERES

**TEXT.** Cadenas de caracteres de “**longitud ilimitada**”. A diferencia de los tipos **CHAR** y **VARCHAR**, este tipo de dato no se puede descomponer en caracteres individuales para su análisis.

### Sintaxis.

Nombre\_campo **TEXT**

### Ejemplo.

Descripcion **TEXT** => ‘**Mesa color caoba oscura, para 8 personas**’

Direccion **TEXT** => ‘**San Rafael Atlixco, numero 186 Col Vicentina C.P.  
09340 México D.F.** ‘

### Nota.

El tamaño limite de este tipo de dato es de 2,147,483,647 caracteres.

## Tipos de datos. NUMERICOS

Tipo	Rango	Espacio para almacenamiento
<b>BIGINT</b>	de 0 a 18.446.744.073.709.551.615 <b>UNSIGNED</b>	<b>8 bytes</b>
<b>INT</b>	-2.147.483.648 a 2.147.483.647 ( <b>SIGNED</b> ); de 0 a 4.294.967.295 ( <b>UNSIGNED</b> );	<b>4 bytes</b>
<b>SMALLINT</b>	de 32.768 a 32.767( <b>SIGNED</b> ); de 0 a 65,535 ( <b>UNSIGNED</b> );	<b>2 bytes</b>
<b>TINYINT</b>	de -128 a 127 ( <b>SIGNED</b> ), de 0 a 255 ( <b>UNSIGNED</b> )	<b>1 byte</b>
<b>BIT</b>	de -128 a 127 ( <b>SIGNED</b> ), de 0 a 255 ( <b>UNSIGNED</b> )	<b>1 byte</b>



## Tipos de datos. NUMERICOS REALES

Tipo	Rango	Espacio para almacenamiento
<b>Double, Real</b>	-1.797 E+308 y -2.225E-308 0 2.225 E-308 y 1.797E+308	4 bytes
<b>Float</b>	-3.402E+38 y -1.175E-38, 0 1.175E-38 y 3.402E+38	8 bytes
<b>Decimal</b>	-1,79769313486231 57E+308 a -2,22507385072014E-308 0	Un numero decimal almacenado como una cadena, con un byte de espacio para cada Caracter.
<b>DEC NUMERIC FIXED</b>	2,2250738585072014E-308 a 1,79769313486231 57E+308	Similar a decimal

## Tipos de datos. NUMERICOS REALES

### Ejemplo 1.

- Creacion de la tabla de registro de produccion de
- liquisos de limpieza de la fabrica "X".

```
CREATE TABLE liquidosLimp  
(  
    CveProd INTEGER, -- Denota el identifiocador unico para cada producto  
    Con_Neto_Lab FLOAT, -- Denota el contenido neto el producto en el Laboratorio  
    Con_Neto_Dist DECIMAL (8,2) -- Denota el contenido neto el producto para el cliente  
);
```

## Tipos de datos. DECIMAL

Permite definir cuantas cifra decimales aparecen después del punto en datos fraccionales.

### Sintaxis:

Nombre\_Campo **DECIMAL** ( N, d )

donde:

**N** es el total de dígitos del dato.

**d** es el numero de cifras decimales que aparecerán en el campo

### Ejemplo

896.25    =>    DECIMAL ( 5,2 )

2003.2569    =>    DECIMAL ( 8,4 )

### Nota:

Si el numero de cifra totales insertadas es mayor que el numero de cifras totales declaradas, el manejador trunca el dato en cuestion.

## Tipos de datos. FECHA Y HORA

Tipo	Descripción	Rango	Formato almacenamiento
<b>Date</b>	almacena una fecha	1 de enero del 1001 al 31 de diciembre de 9999	<b>AAAA-DD-MM</b>
<b>DateTime</b>	Almacena fecha y hora	de enero 1001 a las 0 horas, 0 minutos y 0 segundos a 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos	<b>AAAA-DD-MM HH:MI:SS</b>
<b>Time</b>	almacena un datos de hora	-838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos	<b>HH:MM:SS</b>

## INSERT INTO

**Sintaxis 1:** Esta sintaxis se utiliza cuando se van a escribir datos en **TODOS** los campos de una tabla.

```
INSERT INTO Nom_Tabla VALUES (dato1, dato2, dato3,...);
```

**Nota.**

Debe haber tantos datos como campos en la tabla y los datos deben ser proporcionados en el orden en el que aparecen los campos en la tabla

### Ejemplo 1:

-- Suponiendo que tenemos la declaración de una tabla como la siguiente:

```
CREATE TABLE Ventas (  
    IDPedido INT,  
    ClaveEmpleado INT NOT NULL,  
    Cliente VARCHAR (40) NOT NULL  
);
```

```
INSERT INTO Ventas VALUES (2025, 30, 'Saenz');
```



# INSERT INTO valores en todos los campos

## Sintaxis 1:

**INSERT INTO** Nom\_Tabla (Campo1, Campo2, Campo3,...) **VALUES** (dato1, dato2, dato3,...)

## Ejemplo 1:

Suponiendo que tenemos la declaración de una tabla como la siguiente:

```
CREATE TABLE Aspirantes (IdAspirante INT NOT NULL,  
                           Nombre VARCHAR (35) NOT NULL,  
                           Experiencia TEXT NULL);
```

```
INSERT INTO Aspirantes (IDAspirante, Nombre, Experiencia) VALUES (2025, Karla, 'Ibope  
Administador de proyectos') ;
```

## INSERT INTO valores en algunos campos

**Sintaxis 2:** Esta sintaxis se utiliza cuando **NO** se va a escribir datos en **TODOS** los campos de una tabla.

**INSERT INTO** Nom\_Tabla (*Campo1, Campo2, Campo3,...*) **VALUES** (dato1, dato2, dato3,...)

### Ejemplo 1:

-- Suponiendo que tenemos la declaracion de una tabla como la siguiente:

```
CREATE TABLE Ventas (IDPedido INT PRIMARY KEY,  
                       ClaveEmpleado INT NOT NULL,  
                       Cliente VARCHAR (40) NOT NULL,  
                       FechaVenta DATE NULL);
```

```
INSERT INTO Ventas (IDPedido, ClaveEmpleado, Cliente) VALUES (2025, 30, 'Saenz' );
```

## Constraint “PRIMARY KEY” o Clave primaria

Una **clave primaria** son uno o mas campos que **identifican de manera única** a cada una de las filas de una tabla.

Si la **clave primaria** es un solo campo este debe cumplir con las siguientes condiciones: su valor es **único y no vacíos** para cada fila o registro.

**Sintaxis 1. Clave Primaria simple** - Un solo campo como clave primaria –

```
create table Nom_Tabla (Campo1 TIPO,  
                        Campo2 TIPO,  
                        Campo3 TIPO,  
                        PRIMARY KEY (Campo1))
```

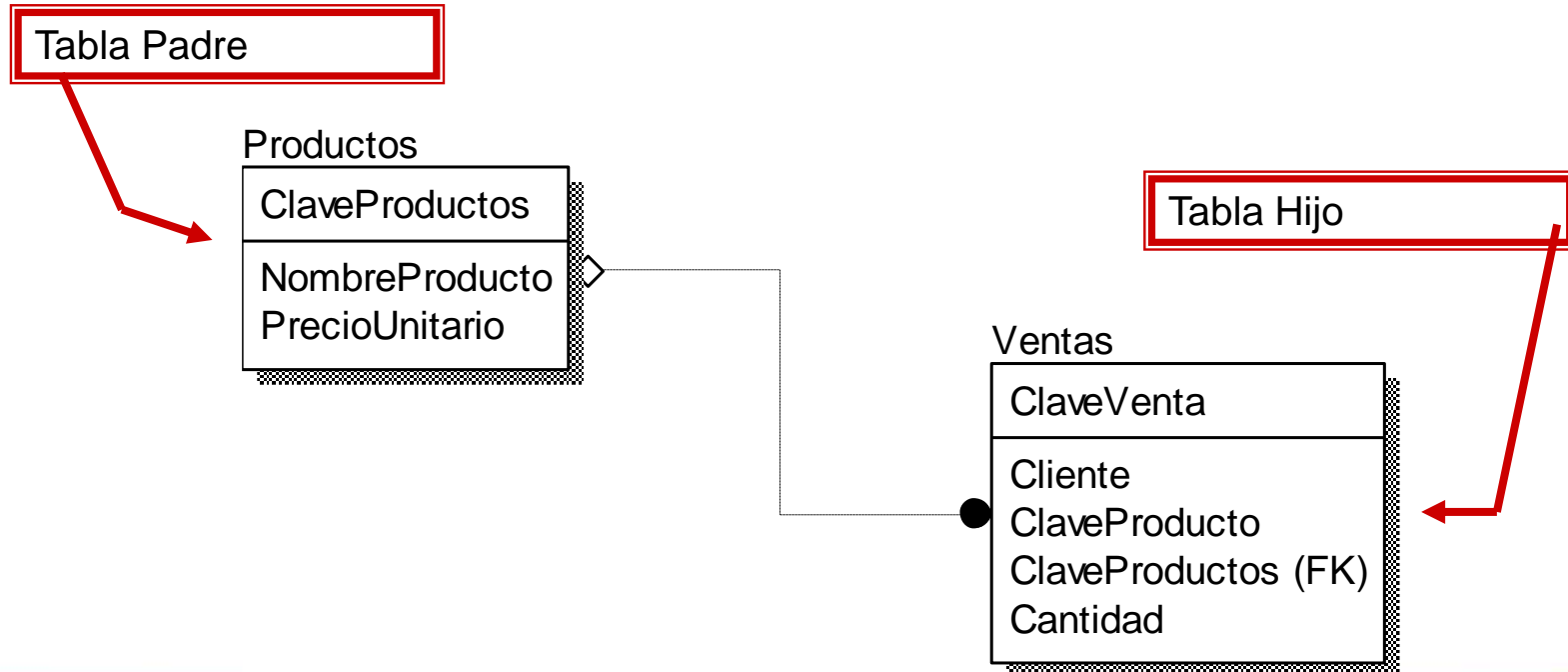
**Sintaxis 2. Clave primaria Compuesta** - Dos o mas campos –

```
create table Nom_Tabla (Campo1 TIPO,  
                        Campo2 TIPO,  
                        Campo3 TIPO,  
                        PRIMARY KEY (Campo1, Campo2))
```

## Definición foreign key

Una Restricción **FOREIGN KEY** o también conocida como **llave externa**, es una columna o combinación de columnas que se utiliza para **establecer y exigir un vínculo entre los datos de dos tablas**.

La restricción **Foreign key** genera un vínculo entre dos tablas, de las cuales a una de ellas se le denomina **Tabla Padre** y a la otra **Tabla Hijo**.



## Constraint Foreign key

**Genera una restricción a partir de la relación de dos tablas.**

En la figura siguiente se muestra un caso de aplicación de una restricción **FOREIGN KEY**, en la cual una venta registrada en la tabla “Ventas” Contiene la clave de un producto que obligadamente debe aparecer en la tabla de productos, con esto nos aseguramos que se realicen ventas de los productos que se ofrecen.

**Productos**

ClaveProducto	NombreProducto	PrecioUnitario
2021	Producto 1	10.22
2022	Producto 2	100.25
2023	Producto 3	10.50
2024	Producto 4	11.00
2025	Producto 5	15.00
2026	Producto 6	89.45
2027	Producto 7	80.00
2028	Producto 8	12.00
2029	Producto 9	25.00

**Ventas**

ClaveVenta	Cliente	ClaveProducto	Cantidad
10025	Cliente 1	2021	5
10025	Cliente 1	2022	6
10025	Cliente 1	2029	78
10026	Cliente 2	2027	45
10027	Cliente 3	2023	1
10028	Cliente 4	2021	2
10029	Cliente 5	2028	3
10029	Cliente 5	2021	45
10029	Cliente 5	2029	71

**Verifica que la clave de producto este en la tabla productos**

## Definición Foreign key

### Sintaxis:

```
CREATE TABLA Tabla_Padre (Campo1 tipoX PRIMARY KEY,  
                           Campo2 tipoY,  
                           Campo3 tipoZ)
```

### Sintaxis

```
CREATE TABLA Tabla_Hijo (Campo11 tipoA,  
                          Campo1 tipoX,  
                          Campo12 tipoB,  
                          FOREIGN KEY (Campo1) REFERENCES T1 (Campo1))
```

### Reglas para generar una **FOREIGN KEY**:

1. La tabla a la que se hace referencia debe existir antes de crear la tabla que contiene la **FOREIGN KEY**. Además debe tener un campo definido como **PRIMARY KEY**
2. Los campo **PRIMARY KEY** en la primera tabla y **FOREIGN KEY** en la segunda tabla, deben ser del **mismo tipo, pero no necesariamente el mismo nombre.**



# **Consultas - Querys**

## **La sentencia SELECT**

## Sentencia SELECT

La sentencia **SELECT** permite consultar la información contenida en una o varias tablas.

### Sintaxis 1.

```
SELECT * FROM Tabla
```

### Donde:

**SELECT**. Indica que se muestra información

**FROM**. Indica de donde provienen esa Información.

Esta sintaxis Muestra Todo el contenido de la tabla mencionada después de la cláusula FROM.

Muestra todas las Columnas y todos los registros contenidos en la tabla

## Sentencia SELECT - Columnas seleccionadas

Si se desea realizar una consulta que muestre determinadas columnas, se debe de seguir la siguiente sintaxis.

### Sintaxis 2.

```
SELECT Columna1, Columna3, Columna 25,...  
FROM Tabla
```

### Donde:

**Columna1, Columna3 y Columna25.** Son Columna de la tabla mencionada después de la cláusula FROM.

Las Columnas no necesariamente son adyacentes.

Pueden listarse solo algunas o todas la columna de la tabla, lo cual seria equivalente a colocar un asterisco (\*)

## Sentencia SELECT - Columnas seleccionadas

**SELECT \* FROM Alumnos**

Matricula	Nom	App	Tel	Direc	Status	Sexo	FecNac
993248	GABRIEL	CERVANTES JIMENEZ	56564714	Via Ludovico il Moro 22	1	M	1982-09-30
993251	FELIPE	RODRIGUEA ACOSTA	15475926	Rue Joseph-Bens 532	1	M	1981-02-02
993254	LORENA	SANCHES VARGAS	14253689	43 rue St. Laurent	1	F	1985-05-23
993257	JUAN CARLOS	CRUZ CRUZ	55663322	Heerstr. 22	1	M	1982-05-29
993260	CARLOS	MORA RAMIRO	55887415	South House 300 Queensbridge	1	M	1982-08-16
993263	IGNACIO	RAMIREZ ALTAMIRANO	59784512	Ing. Gustavo Moncada 8585 P...	1	M	1981-10-10

**SELECT Matricula, Nom, App  
FROM Alumnos**

Matricula	Nom	App
993248	GABRIEL	CERVANTES JIMENEZ
993251	FELIPE	RODRIGUEA ACOSTA
993254	LORENA	SANCHES VARGAS
993257	JUAN CARLOS	CRUZ CRUZ
993260	CARLOS	MORA RAMIRO
993263	IGNACIO	RAMIREZ ALTAMIRANO
993266	JUAN JOSE	ALFARO RODRIGUEZ
993269	MARIA	GOMEZ CEDEDO
993272	SARA	CUMBERAS AGULAR
993275	CARMEN	RAMIREZ DIAZ

## Generando ALIAS para las Columnas

En ocasiones cuando se realiza una consulta hacia una tabla, el encabezado de las columnas no son muy descriptivas, o por el contrario el nombre del encabezado es bastante largo, etc. Cuando se presentan caso como estos, se puede asignar un ALIAS a las columnas de las consulta.

### Sintaxis

```
SELECT ColumnaX AS AliasColumnaX,  
        ColumnaY AS AliasColumnaY  
FROM Tabla
```

Si el alias contiene espacios en blanco se debe de agrupar entre Comillas ( " " ).

### Sintaxis

```
SELECT ColumnaX AS "Alias ColumnaX" ,  
        ColumnaY AS "Alias ColumnaY"  
FROM Tabla
```

## ALIAS para las Columnas - Ejemplo

```
SELECT Nom, app, Sexo  
FROM Alumnos
```

Nom	app	Sexo
GABRIEL	CERVANTES JIMENEZ	M
FELIPE	RODRIGUEA ACOSTA	M
LORENA	SANCHES VARGAS	F
JUAN CARLOS	CRUZ CRUZ	M
CARLOS	MORA RAMIRO	M
IGNACIO	RAMIREZ ALTAMIRANO	M
JUAN JOSE	ALFARO RODRIGUEZ	M

```
SELECT Nom AS Nombre,  
       app AS "Apellidos",  
       Sexo AS "Sexo del paciente"  
FROM Alumnos
```

Nombre	Apellidos	Sexo del paciente
KARINA	CRUZ CRUZ	F
GABRIEL	CERVANTES JIMENEZ	M
FELIPE	RODRIGUEA ACOSTA	M
LORENA	SANCHES VARGAS	F
JUAN CARLOS	CRUZ CRUZ	M
CARLOS	MORA RAMIRO	M
IGNACIO	RAMIREZ ALTAMIRANO	M



## Clausula DISTINCT

Esta sentencia es usada en una consulta para eliminar valores duplicados en una columna.

### Sintaxis

```
SELECT DISTINCT Columna  
FROM Nom_tabla
```

**Ejemplo.** Consulta que muestra únicamente los puestos laborales que puede ocupar un empleado.

```
SELECT DISTINCT title  
FROM Employees
```

```
SELECT title  
FROM Employees
```

### resultado:

title

-----  
Inside SalesCoordinator  
Sales Manager  
Sales Representative  
  
Vice President, Sales

title

-----  
Sales Representative  
Vice President, Sales  
Sales Representative  
Sales Representative  
Sales Manager  
Sales Representative  
Sales Representative



# Gracias!

