

# Curso Sql Básico

---

Profesor: Ignacio Lequerica Navarro  
nacho@jacar.es

# TEMARIO

- Módulo 1: Introducción
- Módulo 2: DML
- Módulo 3: DDL
- Módulo 4: DTL

# HORARIO

Horario	Lunes	Martes
14:00 – 14:30	1. Introducción	3. DML
14:30 – 15:00	1. Introducción y Parte Práctica	3. Parte Práctica
15:00 – 15:30	2. DDL	3. DML
15:30 – 15:50	2. Parte Práctica	3. Parte Práctica
<b>15:50 – 16:10</b>	<b>DESCANSO</b>	<b>DESCANSO</b>
16:10 – 16:30	2. DDL	3. DML
16:30 – 17:00	2. Parte Práctica	3. Parte Práctica
17:00 – 17:30	3. DML	4. DTL
17:30 – 18:00	3. Parte Práctica	4. Parte Práctica

# MODULO 1. Introducción

- ¿Qué es SQL?
- ¿Qué es una base de datos?
- ¿Qué significa una base de datos relacional?
- Tablas, Columnas y filas
- Claves
- Ejemplo y optimización
- Sentencias SQL
- Comandos
- Convención nombres
- Tipos de datos
- RDBMS
- Caso práctico

# MODULO 1. Introducción

## ¿Qué es SQL?

- SQL: Structured Query Language
- Un lenguaje creado específicamente para gestionar bases de datos relacionales
- Lenguaje declarativo (Escribimos lo que queremos hacer) a diferencia del procedimental (Escribimos un procedimiento)
- Originalmente basado en el álgebra relacional y en el cálculo relacional. Creado originalmente en 1970
- ANSI (supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas) desde el 1986 y sigue los estandars de ISO (organización para la creación de estándares internacionales compuesta) desde el 1987
- SQL consiste en un lenguaje de definición de datos (DDL), un lenguaje de manipulación de datos (DML), lenguaje de control de transacciones (TCL) y un lenguaje de control de datos (DCL)

# MODULO 1. Introducción

## ¿Qué es una base de datos?

- Contenedor para organizar la información de una forma constructiva
- Util cuando tenemos mucha información (Imaginar 500 hojas de Excel)
- Centralizando será mas fácil consultar, actualizar, insertar y borrar
- Diferentes tipos de base de datos:
  - Relacional
  - Orientada a objetos
  - Bases de datos documentales nosql (MongoDb...)

# MODULO 1. Introducción

## ¿Qué significa una base de datos relacional?

- Una base de datos relacional está basada en SQL
- Es una forma de describir la información y las relaciones entre entidades
- El modelo relacional es un modelo matemático basado en el álgebra relacional y en el cálculo relacional
- SQL ha ido variando el modelo relacional

# MODULO 1. Introducción

## ¿Qué significa una base de datos relacional?

- Para representar las bases de datos relacionales se utiliza principalmente el modelo entidad-relación.
- Entidad: Representa cosas o objetos
  - Empresa
  - Tipo de dirección
  - Cliente
  - Actividades
- Atributos: Identifican las características de la entidad
  - Empresa: Tiene nombre comercial y página web



# MODULO 1. Introducción

## ¿Qué significa una base de datos relacional?

- Relación:
  - Uno a uno → Una empresa puede ser un cliente
  - Uno a varios → Una empresa tiene varias oficinas
  - Varios a varios → Las empresas pueden tener varias actividades
- Claves:
  - Primaria → Atributo que va a permitir que no se repita esa entidad
  - Clave externa → Atributo de una entidad relacionada con una clave primaria de otra entidad

# MODULO 1. Introducción

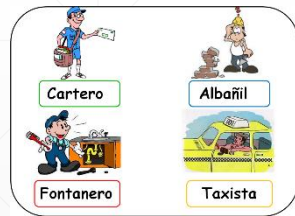
¿Qué significa una base de datos relacional?



Pedidos



Clientes



Actividades



Empresas



Oficinas



Dirección calles

# MODULO 1. Introducción

## Tablas, Columnas y filas

- En una base de datos relacional la información se almacena en una tabla
- Una tabla tiene un nombre y una colección de columnas
- Cada columna tiene un nombre, con restricciones de tamaño, el tipo que se puede almacenar y si es información obligatorio o no
- Cada fila almacenará la información al menos de las columnas obligatorias
- Las filas pueden ser devueltas preguntando acerca de las columnas realizando consultas (Cuales son los clientes que empiezan por A)

# MODULO 1. Introducción

## Claves

- Parte esencial en los modelos relacionales
- Cada tabla tiene que tener una columna única que pueda identificar a la fila, lo que llamamos clave primaria (PRIMARY KEY)
- Una tabla puede tener una clave externa (FOREIGN KEY), que enlaza con la clave primaria de una tabla
- Una clave primaria puede ser natural como un ISBN o un CIF o inventada como una clave autonumerica

# MODULO 1. Introducción

## Ejemplo y optimización

- Queremos almacenar empresas y actividades. Posibilidades de diseño:

NombreFiscal	Actividad1
Experian	Adquisición de clientes e inteligencia de mercado
Jacar Systems	Consultora Informática

- Queremos una actividad mas:

NombreFiscal	Actividad1	Actividad2
Experian	Adquisición de clientes e inteligencia de mercado	Servicios de marketing
Jacar Systems	Consultora Informática	Servicios de marketing

- No es lo más óptimo ya que no es una solución dinámica

# MODULO 1. Introducción

## Ejemplo y optimización

- Solución más optima:

Clave	NombreFiscal
1	Experian
2	Jacar Systems

Clave	ClienteClave	Actividad
1	1	Servicios de marketing
2	1	Adquisición de clientes e inteligencia de mercado
3	2	Consultora Informática
4	2	Servicios de marketing

# MODULO 1. Introducción

## Ejemplo y optimización

- Solución más optima:

Clave	NombreFiscal
1	Experian
2	Jacar Systems

ClaveEmpresa	ClaveActividad
1	1
1	2
2	1
2	3

Clave	Actividad
1	Servicios de marketing
2	Adquisición de clientes e inteligencia de mercado
	Consultora Informática

# MODULO 1. Introducción

## Sentencias SQL

- Sentencia SQL es una combinación de palabras algunas propias de SQL (basadas en inglés) y otras definidas por nosotros
- Las sentencias pueden ser divididas en cláusulas
- Las sentencias terminan con ;
- SQL no discrimina entre mayúsculas y minúsculas (no es case-sensitive)
- Comentarios:
  - Para una línea: --
  - Para muchas líneas: /\* \*/

*SELECT VALUES FROM TABLENAME;*

`SELECT MyColumnName, 'Constant' FROM MyTableName;`



# MODULO 1. Introducción

## Sentencias SQL

- Ejemplos:

*SELECT VALUES FROM TABLENAME;*      *SELECT NombreFiscal FROM Empresas;*

- Para devolver todas las columnas utilizamos el comodín \*
- Ejemplos para devolver todas las columnas:

*SELECT \* FROM Empresas;*

# MODULO 1. Introducción

## Convención nombres

- Utilizar una convención de nombre es importante para seguir un estándar.
- No hay un estándar común
- En este curso vamos a utilizar:
  - Tablas en nombre plural
  - Claves primarias siguen el formato Nombre\_Tabla\_Singular + Id
  - Notación PascalCase
  - Columnas en singular
  - Las palabras reservadas de SQL en mayúsculas
  - Para el nombre de las restricciones de claves primarias utilizamos el formato PK\_Empresald
  - Para el nombre de las restricciones de claves externas utilizamos el formato FK.TablaPrincipal\_TablaExterna\_ColumnaClaveExterna

# MODULO 1. Introducción

## Tipos de datos

Tipo	Valor
Caracter	Puede almacenar N caracteres de manera estatica
Varchar	Puede almacenar N caracteres de manera dinámica
Binary	Información hexadecimal
SmallInt	$-2^{15}$ (-32,768) to $2^{15}-1$ (32,767)
Integer	$-2^{31}$ (-2,147,483,648) to $2^{31}-1$ (2,147,483,647)
BigInt	$-2^{63}$ (-9,223,372,036,854,775,808) to $2^{63}-1$ (9,223,372,036,854,775,807)
Boolean	True o false
Date	Formato YYYY-MM-DD
Time	Formato HH:MM:SS
TimeStamp	Ambos Date y Time

# MODULO 1. Introducción

## RDBMS

- Relational Database Managament System
- Extienden ANSI SQL con extensiones propias del vendedor.
- Oracle → PL/SQL
- SQL Server → T-SQL
- ANSI SQL funcionará en cualquier RDBMS

# MODULO 1. Introducción

## Caso Práctico

- Vistazo general herramienta sql server managment studio para poder realizar una base de datos, tablas, consultas, insertar, borrar, actualizar y crear relaciones

# MODULO 1. Introducción

## Caso Práctico

- Realizar Modulo1\_Ejercicio1

# MODULO 1. Introducción

## Resumen

- SQL es el lenguaje que nos va a permitir gestionar bases de datos relacionales
- Las bases de datos relacionales van a contener información almacenadas en formas de tabla y relaciones entre ellas

# MODULO 2. DDL

- ¿Qué es DDL?
- Crear base de datos
- Crear tabla
- Columnas autonuméricas
- Clave Primaria
- Caso práctico
- Restricciones
- Modificar tabla
- Borrar tabla
- Caso práctico



# MODULO 2. DDL

## ¿Qué es DDL?

- DDL: Data Definition Language
- Comandos para crear y modificar construcciones en la base de datos
- La mayoría de RDBMS tienen herramientas para hacer esto de manera más sencilla, pero vamos a explicar las sentencias para entenderlo mejor.

# MODULO 2. DDL

## Crear base de datos

- No es ANSI
- Sentencia:
  - *--this is not ANSI SQL*
  - *--but is supported by most vendors*
  - *CREATE DATABASE Experian;*
  - *USE DATABASE Experian;*

# MODULO 2. DDL

## Crear tabla

- Es ANSI
- Sentencia:

```
CREATE TABLE Empresas(Empresald INTEGER,  
NombreComercial VARCHAR(300));
```

- Podemos especificar si la columna es nula o no nula (si es obligatoria)
- NULL es el valor por defecto
- Si intentamos insertar un valor nulo en un columna no nula nos dará error

```
CREATE TABLE Empresas(Empresald INTEGER NOT NULL,  
NombreComercial VARCHAR(300) NOT NULL);
```

# MODULO 2. DDL

## Identity

- `IDENTITY [ (seed, increment) ]`
- Para las columnas clave primaria autonuméricas utilizamos la siguiente sentencia:

```
CREATE TABLE Empresas(Empresald INTEGER IDENTITY(1,1) PRIMARY KEY,  
NombreComercial VARCHAR(300));
```

# MODULO 2. DDL

## Clave Primaria

- Por defecto las claves primarias son no nulas
- En una tabla más de una columna pueden ser clave primaria

*CREATE TABLE Empresas(Empresald INTEGER PRIMARY KEY,  
NombreComercial VARCHAR(300));*

# MODULO 2. DDL

## Restricciones

- La palabra clave en SQL es CONSTRAINT
- Lo podemos utilizar para crear una clave primaria también
- Se utiliza para crear las claves externas (FOREIGN KEY)
- Se añaden al final de la definición de columnas

```
CREATE TABLE Empresas(Empresald INTEGER,NombreComercial VARCHAR(300) CONSTRAINT  
PK_Empresald PRIMARY KEY(Empresald));
```

# MODULO 2. DDL

## Caso Práctico

- Vamos a realizar el Modulo2\_Ejercicio1

# MODULO 2. DDL

## Modificar tabla

- Nos permite modificar una columna o una CONSTRAINT en una tabla existente
- Se utiliza ALTER TABLE
- Lo podemos utilizar para crear una clave primaria también
- Se utiliza para crear las claves externas (FOREIGN KEY)
- Se añaden al final de la definición de columnas

```
ALTER TABLE Oficinas ADD CONSTRAINT FK_Oficinas_Empresas_Empresald FOREIGN  
KEY(Empresald) REFERENCES Empresas(Empresald);
```



# MODULO 2. DDL

## Drop table

- Para borrar las tablas utilizamos la palabra reservada DROP TABLE
- También borrará su contenido
- No podremos borrar una tabla si otra tabla tiene una relación de clave externa con ella

DROP TABLE empresas;

# MODULO 2. DDL

## Resumen

- CREATE TABLE para crear tablas
- ALTER TABLE para modificarlas
- DROP TABLE para borrarlas
- CONSTRAINT para crear relaciones de clave primaria y clave externa
- IDENTITY para definir columnas autonumericas

# MODULO 2. DDL

## Caso Práctico

- Vamos a realizar el ejercicio Modulo2\_2 y Modulo2\_3

# MODULO 3. DML

- ¿Qué es DML?
- SELECT
- WHERE
- AND y OR
- BETWEEN
- LIKE
- IN
- IS
- IS NOT
- Caso práctico

# MODULO 3. DML

## ¿Qué es DML?

- DML: Data Modeling Language
- Estas son las principales sentencias:
  - SELECT → Consulta de tabla/s
  - INSERT → Insertar registros a tabla
  - UPDATE → Actualizar registros de tabla
  - DELETE → Borrar registros de tabla

# MODULO 3. DML

## SELECT

- Nos va a permitir responder a preguntas sobre la información
- Sentencia:

*SELECT nombre\_de\_columna FROM nombre\_de\_tabla;*

- Si utilizamos el comodín \*, nos devolverá todas las columnas
- Es comodo, pero poco eficiente ya que la base de datos tendrá que mirar todas las columnas y devolverá todas.
- Sentencia:

*SELECT \* FROM nombre\_de\_tabla;*

# MODULO 3. DML

## SELECT

- Como el nombre de columnas se puede repetir en otras tablas, para evitar colisiones pondremos la tabla antes del nombre de columna.
- Sentencia:

*SELECT nombre\_tabla.nombre\_de\_columna FROM nombre\_de\_tabla;*

- Para evitar nombres muy largos utilizamos alias que se declaran con un nombre más corto detrás de la tabla
- Sentencia:

*SELECT p.nombre\_de\_columna FROM nombre\_de\_tabla p;*

# MODULO 3. DML

## SELECT

- Por defecto la clausula SELECT devuelve todos los resultados (ALL)
- Para acotar los resultados hay dos formas:
  - Añadir clausulas después de FROM
  - Incluir DISTINCT
- DISTINCT va a devolver los registros no duplicados
- Sentencia:

*SELECT DISTINCT p.nombre\_de\_columna FROM nombre\_de\_tabla p;*



# MODULO 3. DML

## WHERE

- WHERE actua como un buscador dentro de nuestros resultados
- El contenido de WHERE serán expresiones que serán evaluadas como verdad o falso
- Si estas filas cumplen la condición serán devueltas
- Sentencia:

*SELECT e.NombreComercial FROM Empresas e WHERE e.NombreComercial = 'Pepe';*

# MODULO 3. DML

## WHERE

- Las comparaciones dentro de la clausula WHERE pueden realizarse con los siguientes operadores:

Operador	Operación
=	Igual
<>	Distinto
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

# MODULO 3. DML

## AND y OR

- Las expresiones se pueden combinar con otras expresiones booleanas
- Si dos expresiones las anidamos con AND en el resultado devuelto se tendrán que cumplir las dos.
- Si dos expresiones las anidamos con OR en el resultado devuelto se tendrán que cumplir una de las dos.

```
SELECT e.NombreComercial FROM Empresas e WHERE e.NombreComercial = 'Pepe' AND  
e.PaginaWeb = 'pepe.com';
```

```
SELECT e.NombreComercial FROM Empresas e WHERE e.NombreComercial = 'Pepe' OR  
e.PaginaWeb = 'pepe.com';
```

# MODULO 3. DML

## BETWEEN

- BETWEEN es un operador que devuelve los valores que cumplen entre ese rango superior y inferior
- Sentencia:

*SELECT o.Numero FROM Oficinas o WHERE o.Numero BETWEEN 4 AND 8;*

# MODULO 3. DML

## LIKE

- LIKE es un operador para buscar patrones dentro de cadenas
- Se utiliza con el carácter comodín % que indica cualquiera y puede ir en cualquier parte de la cadena
- Sentencia

*-- Buscar todas las empresas que contengan la e en nombre comercial*

```
SELECT e.NombreComercial FROM Empresas e WHERE e.NombreComercial LIKE '%e%';
```

*-- Buscar todas las empresas que empiezan por la e en nombre comercial*

```
SELECT e.NombreComercial FROM Empresas e WHERE e.NombreComercial LIKE 'e%';
```

# MODULO 3. DML

## IN

- IN es un operador para indicar un conjunto de valores. Devolverá las filas que incluyan los valores indicados en IN
- Sentencia

*-- Buscara todas las empresas que contengan la e en nombre comercial*

*SELECT e.NombreComercial FROM Empresas e WHERE e.NombreComercial IN ('Pepe', 'Pepe2', 'Pepe3');*

# MODULO 3. DML

## IS

- IS es un operador para evaluar los campos que son nulos
- Sentencia

*SELECT e.NombreComercial FROM Empresas e WHERE e.PaginaWeb IS NULL;*

# MODULO 3. DML

## IS NOT

- IS NOT es un operador para evaluar los campos que no son nulos
- Sentencia

*SELECT e.NombreComercial FROM Empresas e WHERE e.PaginaWeb IS NOT NULL;*



# MODULO 3. DML

## RESUMEN

- El comando SELECT es muy poderoso
- Combinando con OR y AND se pueden realizar consultas muy potentes

# MODULO 3. DML

## Caso Práctico

Realizar el ejercicio Modulo2\_2 y Modulo2\_3