Base de datos 1

ACID

- **Atomicity** - Asegurar que la transacción se realice o no, sin quedar a medias ante fallos.
- **Consistency** - Asegurar el estado de validez de los datos en todo momento
- **Isolation** - Asegurar independencia entre transacciones
- **Durability** - Asegurar la persistencia de la transacción ante cualquier fallo.

El concepto de visión es ocultar depende de la accesibilidad del usuario

Nivel Físico - Describe cómo se almacenan realmente los datos.
Nivel Lógico - Describe qué datos se almacenan en la base de datos.
Nivel de vistas - El nivel más alto de abstracción, describe sólo parte de la base de datos completa.

Transacciones
Una transacción es una colección de operaciones que se lleva a cabo como una única función lógica en una aplicación de bases de datos.

Estados de la transacción:
1-**Activa** (estado inicial): la transacción permanece en este estado durante su ejecución.
2-**Parcialmente Comprometida:** la transacción pasa a este estado cuando acaba de realizar la última instrucción.
3-**Fallida:** la transacción pasa a este estado tras descubrir que no puede continuar la ejecución normal.
4-**Abortada:** la transacción pasa a este estado después de haber restablecido la base de datos a su estado anterior.
5-**Comprometida**: la transacción pasa a este estado tras completarse con éxito.

si sale un back vuelve al estado que estaba antes del inicio

todas las bases de datos tienen dos archivos fundamentales, log y archivo principal, log:

gestor de almacenamiento: se va a encargar

las bases de datos tienen varios gestores

UNIDAD 2

Modelo conceptual

Modelos de datos: herramientas para describir los datos
Modelos:
conceptual, vamos a validar el resultado
entidad relación, ordena los datos en tablas conocidas como relaciones
lógicos relacional, entidad relación, capta
físico,

-Diagrama entidad relación(DER):

Tipo de diagrama de flujo que ilustra cómo las "entidades", como se relacionan entre sí dentro de un sistema

Se utiliza un conjunto definido de símbolos, tales como rectángulos, diamantes, óvalos y líneas de conexión para representar la interconexión de entidades, relaciones y sus atributos

Entidades:

fuertes: son independientes de cualquier otra tienen atributos propios,

débiles: derivan su existencia de otra entidad.

Claves de entidad: se refiere a un atributo que únicamente define una entidad en un conjunto de entidades.

Las claves de entidad se dividen en

- **Superclave:** un conjunto de atributos (uno o más) que juntos definen una entidad en un conjunto de entidades.
- **Clave candidata:** es una superclave mínima, es decir, contiene el menor número posible de atributos para seguir siendo una superclave. Un conjunto de entidades puede tener más de una clave candidata.
- **Clave primaria:** es una clave candidata seleccionada por el diseñador de la base de datos para identificar únicamente al conjunto de entidades.

atributos:

simples

derivables

compuestos

multivalorados

los atributos más importantes son:

primary key

foreign key

relaciones, cardinalidad

--

pasando del DER al DLR: las entidades se transforman en tablas y los atributos en campos

DLR

El modelo relacional ordena los datos en tablas, también conocidas como relaciones, cada una de las cuales se compone de columnas y filas

El modelo relacional es un **modelo lógico** que solo sirve para SGBD relacionales

El diagrama lógico relacional (DLR) está compuesto por los siguientes objetos:

Tablas

Campos

Relaciones

EN UNA RELACIÓN DE 1 A 1 LAS ENTIDADES SE HACEN EN UNA SOLA TABLA.

En una tabla del DLR a la entidad que tiene la N le podiamos como un atributo más el atributo identificatorio de la entidad que tiene el 1 ,
la tabla que tiene la N es que va a heredar la FK.
El atributo multivalorado genera una nueva tabla y se va a trabajar como si fuera una relación de 1 a N, en la cual va a recibir como FK el identificador de la tabla en la cual estaba relacionado y va a tener un atributo identificador.
En una relación N a M como mínimo vamos a generar 3 tablas, dos tablas donde está la relación muchos a muchos y luego una tabla intermedia que va a heredar la FK del caso.

Si la relación entre 2 entidades es **N a M**, se generan **3 tablas**:

- Tablas de las entidades: quedan igual, con los mismos campos que atributos tenían en el DER

- Tabla de la relación: lleva el nombre que aparece en el rombo, y tiene relación con las otras tablas a través de las **claves foráneas**

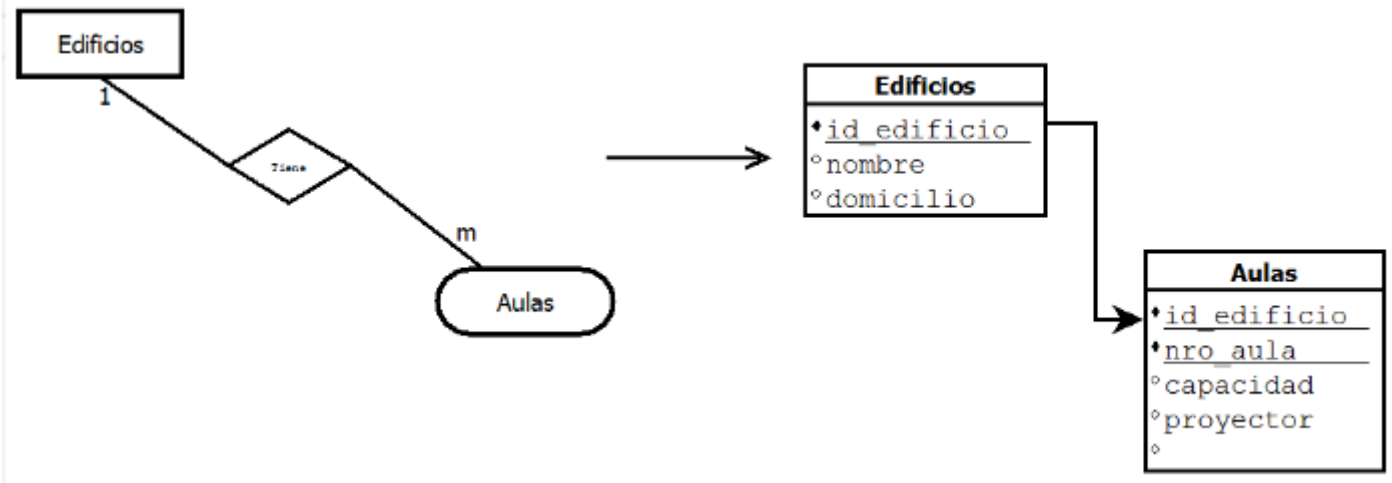
TIPOS DE DATOS
STRING O VARCHAR
INTEGER
DATE O DATETIME
BIT O BOOLEAN

Reglas para el Pasaje

Las entidades se transforman en tablas y los atributo en campos

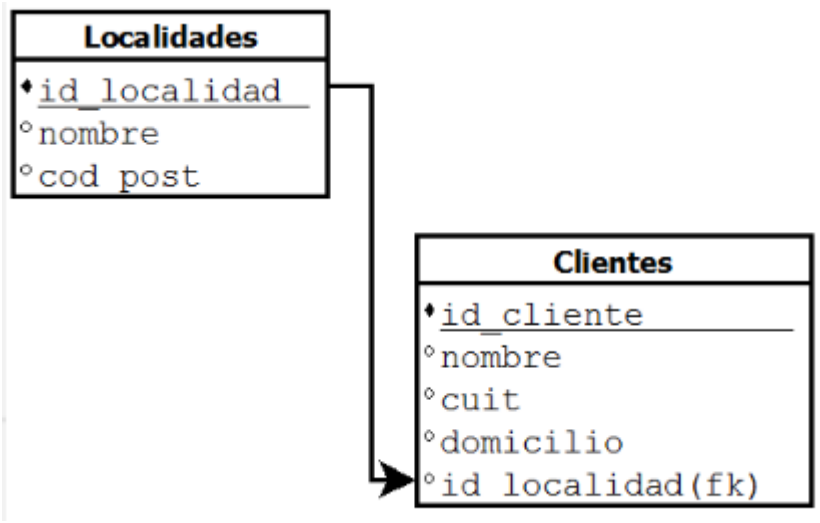
Implicación de Claves

Propagación de claves de entidades fuertes a débiles: la clave propagada es parte de la clave primaria de la entidad débil

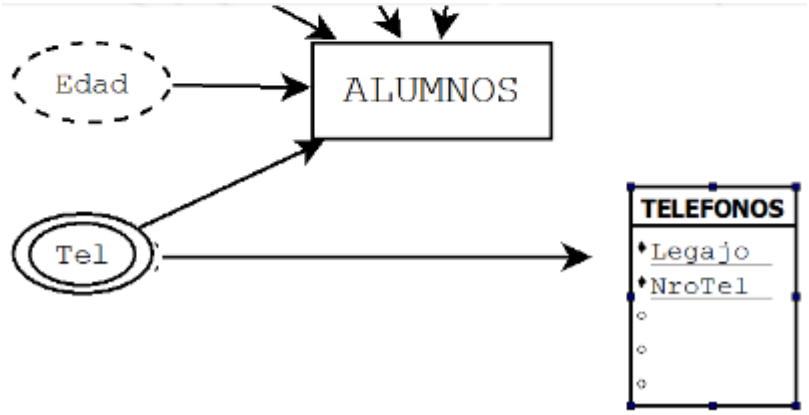


Propagación de claves

De entidad fuerte a entidad fuerte se propaga la clave pero la clave propagada no forma parte de la clave primaria

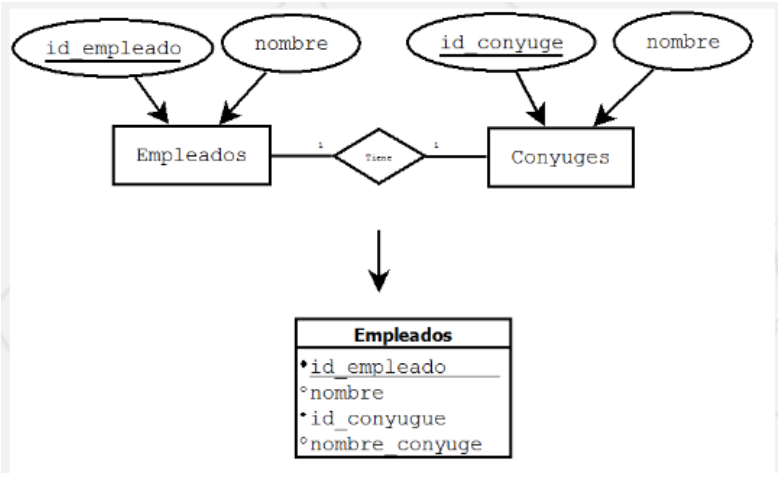


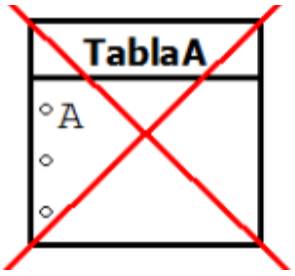
Los atributos multi-valorados se convierten en tablas débiles

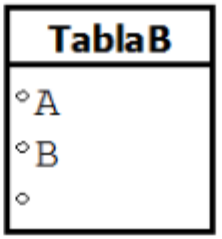


Agregación de Datos: En una relación con cardinalidad 1 a 1 se pueden incluir ambas entidades en una única tabla

Supresión: Si hay dos tablas pero una está incluida en la otra se suprime la primera

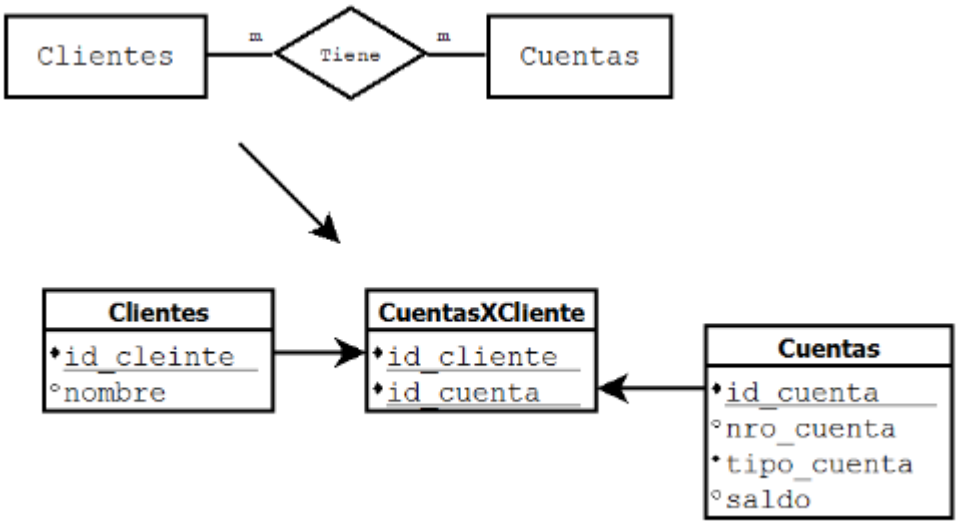






Asociación

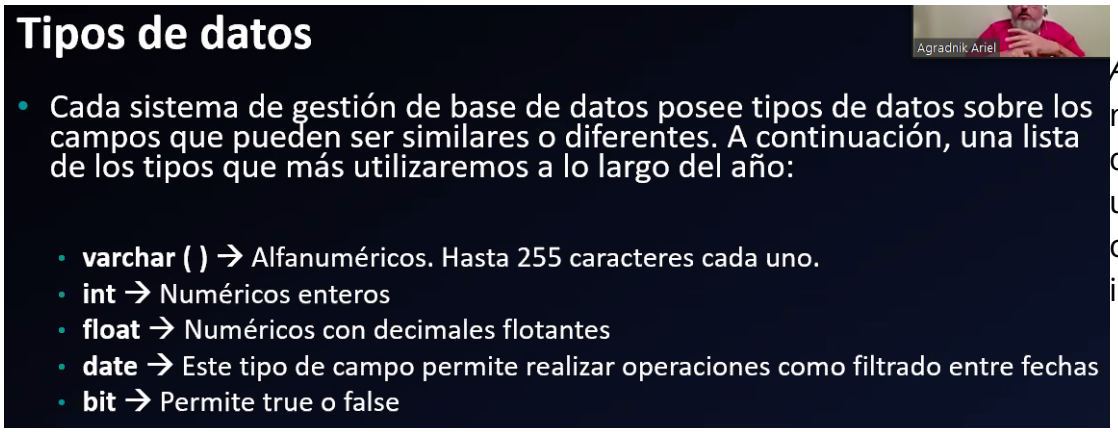
Toda relación con cardinalidad muchos a muchos se traduce en la utilización de una tabla intermedia



Unidad 3

SQL SERVER

Modelo Cliente-Servidor; **sql** es un lenguaje de consultas estructurado



-Campo Autoincremental (ID): números positivos que crecen en una unidad su valor por cada registro incorporado a la tabla

Lenguaje SQL : lenguaje de consultas estructurado (lenguaje ANSI SQL).
tiene 3 sub Lenguajes; **DDL**(nos permite crear la base, las tablas, "create"), **DML**(insertar, de manipulación "select update, delete" manipulamos datos), SPL(bases de datos II)

-Lenguaje de definición de datos **(DDL)**: Las sentencias DDL se utilizan para crear y modificar la estructura de las tablas en la BD

-Las sentencias **DDL:** *clase 27/04*

- **CREATE** - para crear objetos en la base de datos.
- **ALTER** - modifica la estructura de la base de datos.
- **DROP** - borra objetos de la base de datos.

• **TRUNCATE** - elimina todos los registros de la tabla, incluyendo espacios asignados a los registros. Además, reinicia los campos autonuméricos.

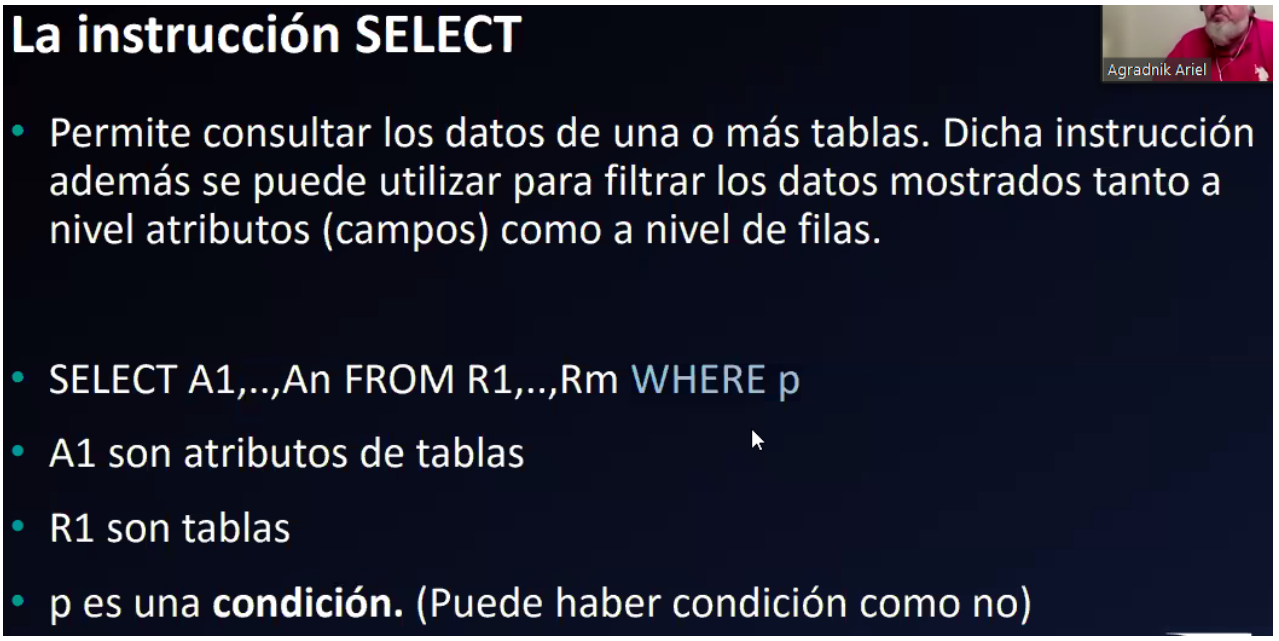
-Lenguaje de manipulación de datos **(DML):** Las sentencias DML son utilizadas para gestionar datos dentro de las bases de datos

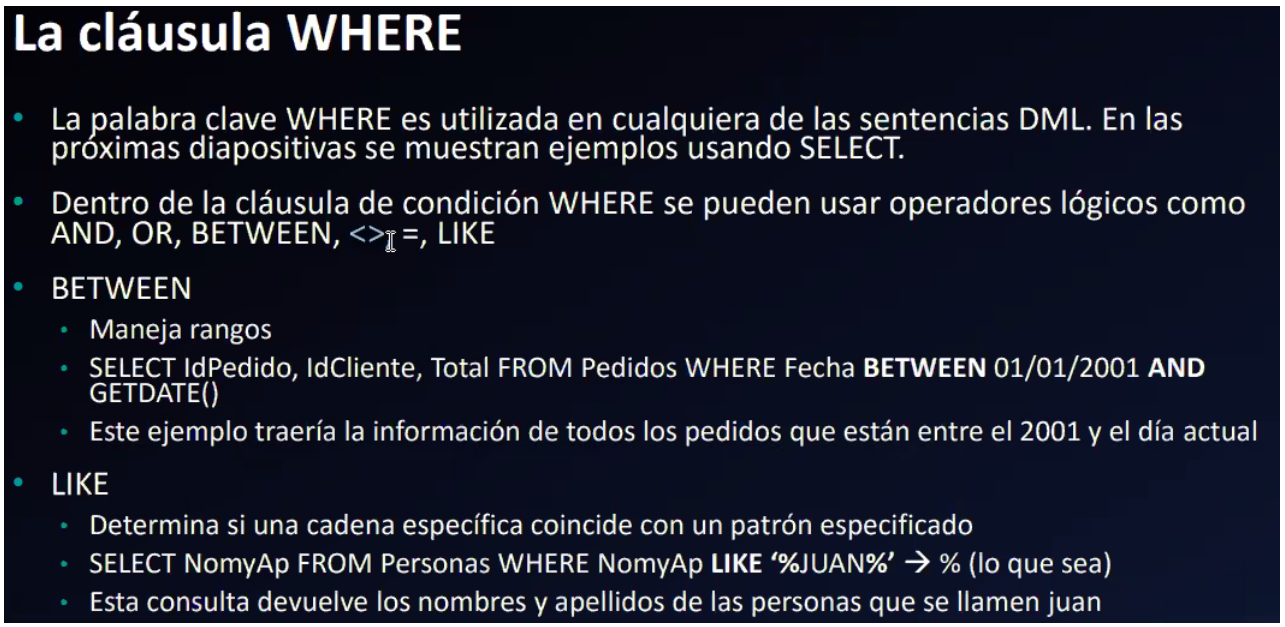
• **SELECT** - para obtener datos de una base de datos. recupera las filas y habilita la selección de varias filas y columnas de tablas que se encuentran contenidas dentro de una BD

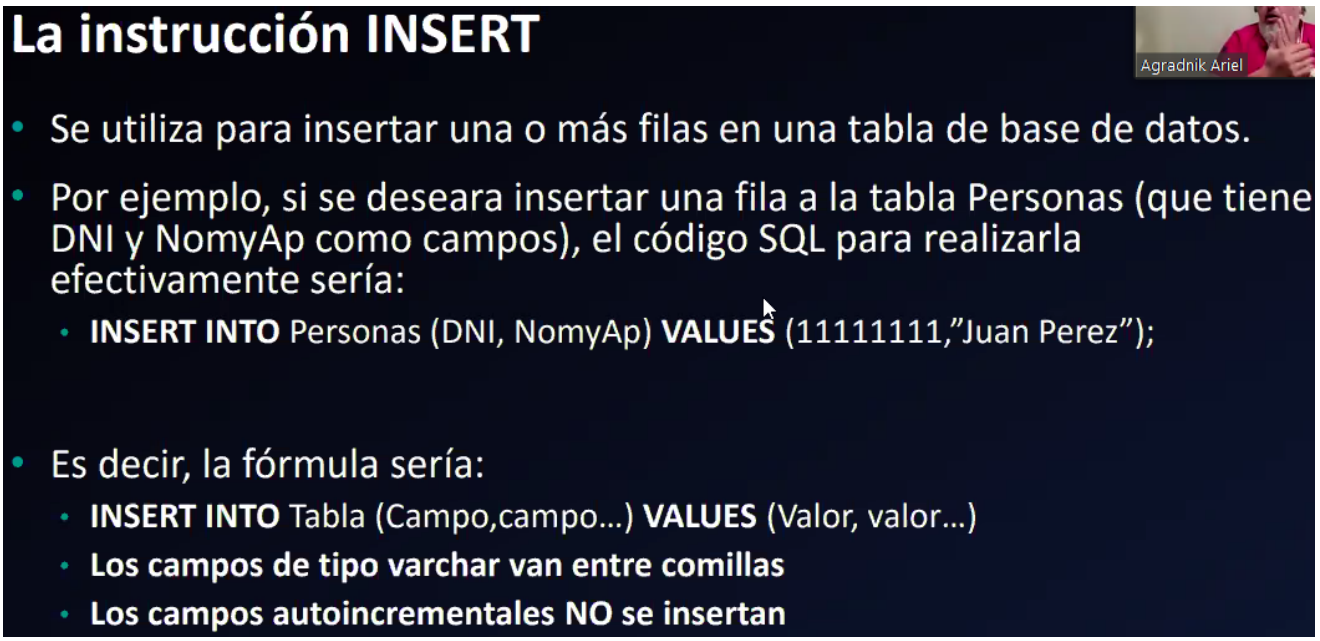
• **INSERT** - para insertar datos a una tabla.

• **UPDATE** - para modificar datos existentes dentro de una tabla.

• **DELETE** - elimina todos los registros de la tabla; no borra los espacios asignados a los registros







SQL avanzado

top si no ponemos order by se ordena según lo agregaron a bd

distinct

funciones de agregación max y min ---- MAX(campo que queremos traer) AS (as es como para renombrar una columna)

in lo que hace es buscar dentro de una bolsa si el campo que estamos buscando se encuentra dentro

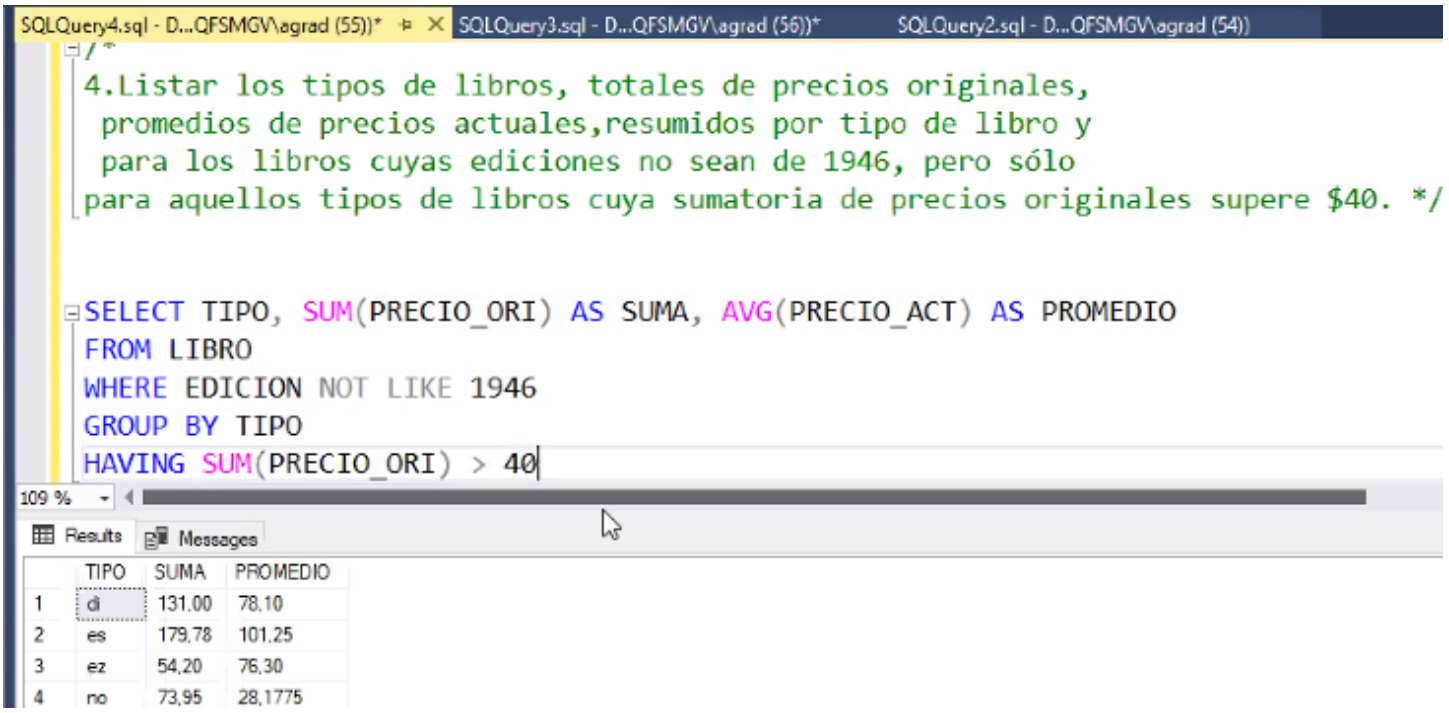
```
select titulo, precio_ori, edicion
from libro where
edicion in ( select anios from tablaAnios)
```

valores nulos
para saber si es nulo IS NULL o IS NOT NULL

dentro de un select se pueden hacer operaciones aritméticas

GROUP BY
HAVING COUNT

new query

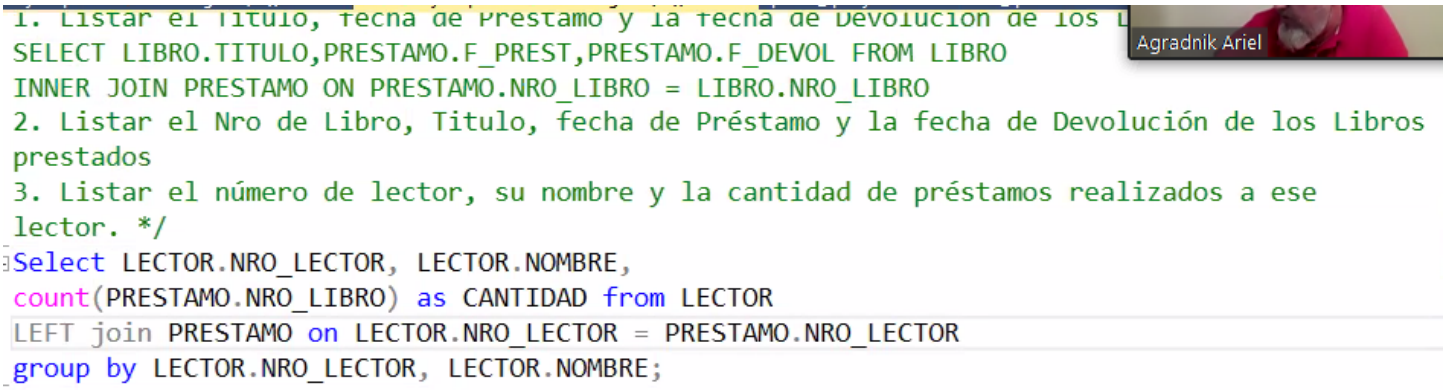


hay 3 formas de hacer las relaciones entre tablas

inner join: tabla1 on tabla1.nombre = tabla2.nombre

podemos tener muchos inner join y left etc

```
SELECT LIBRO.TITULO ,LIBRO.NRO_LIBRO,COPIAS.NRO_COPIA,  PRESTAMO.F_PREST
FROM COPIAS
LEFT JOIN PRESTAMO
     ON PRESTAMO.NRO_COPIA = COPIAS.NRO_COPIA
    AND PRESTAMO.NRO_LIBRO = COPIAS.NRO_LIBRO
INNER JOIN LIBRO ON LIBRO.NRO_LIBRO = COPIAS.NRO_LIBRO
```



4

```
select LIBRO.NRO_LIBRO,LIBRO.TITULO,COPIAS.NRO_COPIa,COUNT(*)  as cantidad_prestmao
from LIBRO
inner join COPIAS on COPIAS.NRO_LIBRO = LIBRO.NRO_LIBRO
inner join PRESTAMO on PRESTAMO.NRO_LIBRO = COPIAS.NRO_LIBRO and PRESTAMO.NRO_COPIA=COPIAS.NRO_COPIA
group by LIBRO.NRO_LIBRO,LIBRO.TITULO,COPIAS.NRO_COPIa
```

```
select *
from LIBRO
inner join COPIAS on COPIAS.NRO_LIBRO = LIBRO.NRO_LIBRO
inner join PRESTAMO on PRESTAMO.NRO_COPIA = COPIAS.NRO_COPIA and PRESTAMO.NRO_LIBRO = COPIAS.NRO_LIBRO
inner join TIPOLIBRO on TIPOLIBRO.TIPO = LIBRO.TIPO
where TIPOLIBRO.DESCTIPO like '%n%' and PRESTAMO.F_DEVOL is null
```

SELECT [**ALL** | **DISTINCT**]

<nombre_campo> [{,<nombre_campo>}]

FROM <nombre_tabla>|<nombre_vista>

[{,<nombre_tabla>|<nombre_vista>}]

[**WHERE** <condicion> [{ **AND** | **OR** <condicion>}]]

[**GROUP BY** <nombre_campo> [{,<nombre_campo >}]]

[**HAVING** <condicion>[{ AND|OR <condicion>}]]

[**ORDER BY** <nombre_campo>|<indice_campo> [**ASC** | **DESC**]

[{,<nombre_campo>|<indice_campo> [**ASC** | **DESC**]}]]

Especificaciones de la sentencia

SELECT: Palabra clave y reservada 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 no repetidos.

FROM: Indica la tabla desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se hace una combinación de tablas usando la instrucción 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. Admiten 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 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.

Funciones

- **ABC()** : para mostrar resultados cantidades que no sean negativas para obtener X campo como un valor absoluto SELECT idCliente, ABS(monto_disponible)
- **TOP** se utiliza para especificar el número de filas a mostrar en el resultado. Esta cláusula **SQL** TOP es útil en tablas con muchos registros, para limitar el número de filas a mostrar en la consulta, y así sea más rápida la consulta, consumiendo también menos recursos en el sistema.
- **Cuando debo usar el WHERE? :**Se debe utilizar para establecer condiciones con los campos que fueron mencionados después del SELECT los cuales no están bajo una función de grupo, dígase COUNT(), SUM(), AVG() y MAX().

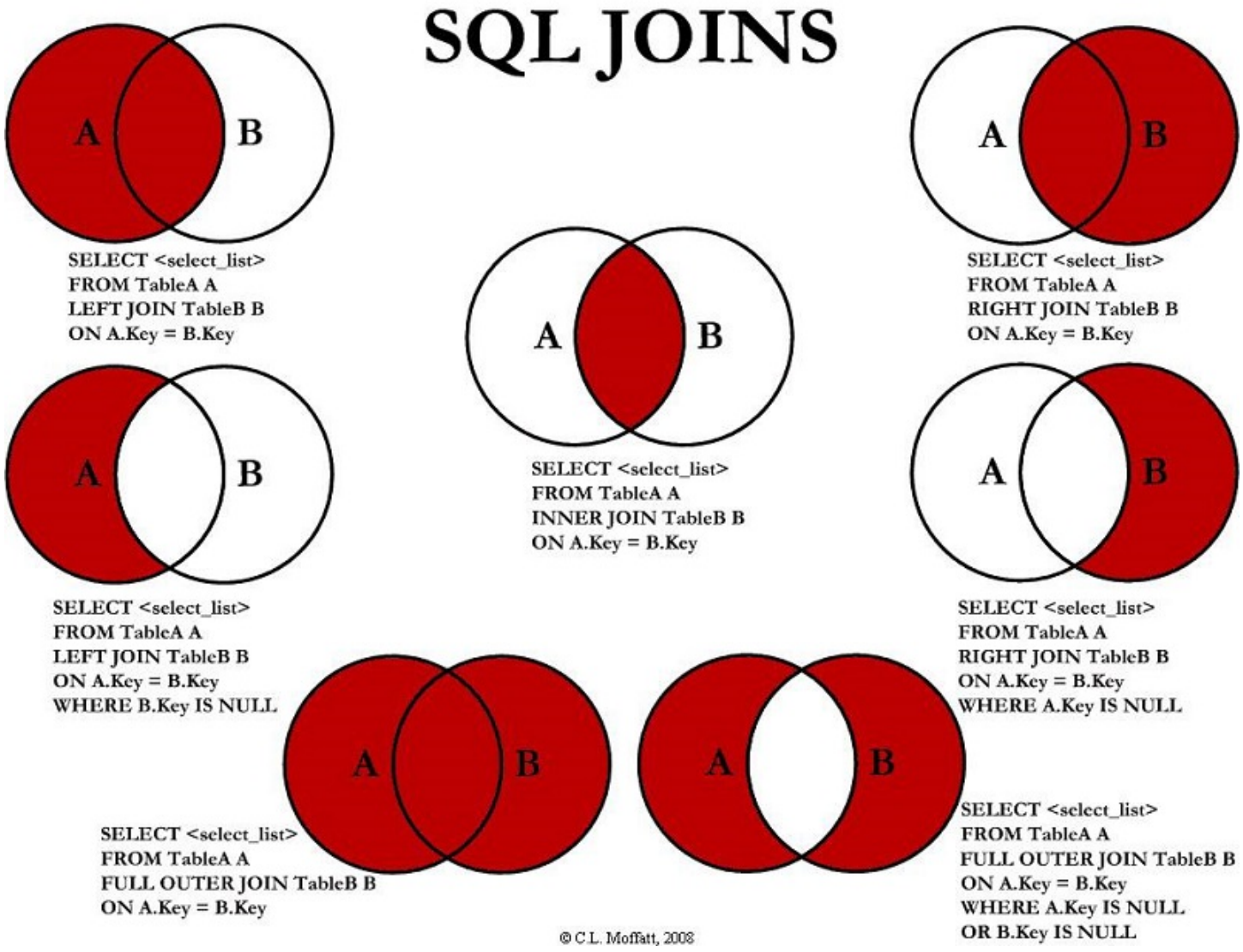
  Se pueden usar en una instrucción "select" y combinarlas con la cláusula "group by". Todas estas funciones retornan "null" si ningún registro cumple con la condición de "where", excepto "count" que en tal caso retorna cero.

- **Count :** se puede emplear con cualquier tipo de dato. devuelve el número total de valores en el campo especificado
- **Min** : con cualquier tipo de dato. devuelve el valor más pequeño en el campo de tabla especificado .
- **Max :** con cualquier tipo de dato. devuelve el valor más grande desde el campo de la tabla especificada .
- **Sum :** sólo en campos de tipo numérico. retorna la suma de los valores que contiene el campo especificado

- **Avg :** sólo en campos de tipo numerico.retorna el valor promedio de los valores del campo especificado
- top
- **DISTINCT :** que nos permite omitir los duplicados de nuestros resultados.
- **¿Cuándo debo usar HAVING? :** Este se debe utilizar cuando necesite establecer condiciones con campos que se encuentren debajo de una función de grupo.
- **Join :** se usa para enlazar los datos de dos tablas relacionadas a través de algún campo en común (típicamente una "foreign key" o clave foránea) y así dar como resultado filas que mezclan datos provenientes de las dos (o más) tablas sobre las que hemos hecho el Join.
- **Inner Join :** se aplica cuando no indicamos otra cosa al hacer la consulta. Devuelve sólo aquellas filas donde haya un "match", es decir, las filas donde el valor del campo de la tabla A que se utiliza para hacer el Join coincida con el valor del campo correspondiente en la tabla B. Ejemplo: devolver todos los productos para los que haya como mínimo un pedido en los últimos días (el inner join enlazará el campo producto en la tabla Pedido con la clave primaria de ese producto en la tabla Producto).
- **Left outer Join.** Cuándo quieres todas las filas para las que haya match pero también aquellas de la Tabla A que no hagan match. Siguiendo el ejemplo anterior, si quieres listar todos los productos con datos de sus pedidos pero mostrando también aquellos productos para lo que no tengas todavía un pedido, la solución sería hacer una Left Outer join entre Producto y Pedido.
- **Right outer Join.** Exactamente lo mismo pero a la inversa, cuando quieres listar las filas de la tabla B aunque no estén relacionadas con ninguna fila de la tabla A. Es un operador un poco redundante ya que se podría cambiar simplemente el orden de las tablas en el Join y utilizar un left outer para conseguir el mismo efecto. No obstante, y como parte de Joins múltiples, es útil tener los dos para un mejor comprensión de la consulta
- **Full outer join.** Es como la suma de las dos anteriores. Queremos tanto las filas de la A como las de B, tanto si hay match como si no (evidentemente cuando haya match la consulta devolverá todos los campos de A y B que hayamos indicado, cuando no, la consulta devolverá sólo los campos de A o B).

Las operaciones aritméticas que implican a **NULL** siempre **devuelve NULL,** por ejemplo, 69 + NULL = NULL.

- ***"IS NULL"*** es la palabra clave que realiza la comparación booleana. Devuelve verdadero si el valor proporcionado es NULO y falso si el valor proporcionado no es NULO.
- ***"NOT NULL"*** es la palabra clave que realiza la comparación booleana. Devuelve verdadero si el valor proporcionado no es NULO y falso si el valor proporcionado es nulo
-



SIMULACRO

Una primary key es…
b. El campo que identifica la tabla y no puede repetirse

El lenguaje SQL está compuesto por…
a. El lenguaje SQL está compuesto por DML, DDL Y SPL

De la siguiente columna, marque cuál considera que es una DDL
a. **CREATE** TABLE Clientes

El nivel físico de abstracción

a. Describe cómo se almacenan realmente los datos.

El DER es un diagrama compuesto por

d. Entidades, relaciones y su cardinalidad

Los modelos de datos son

Respuesta = una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia

Que podemos decir acerca del resultado del siguiente SQL:
SELECT Clientes.CodCliente,Clientes.Nombre,Compra.Fecha FROM Compra LEFT JOIN Clientes ON Clientes.CodCliente = Compra.CodCliente

Respuesta = Devolverá tantas líneas como registros haya en la tabla de compras

La definición de DATO es

Respuesta = Representación de hechos, conceptos, o instrucciones, a la que se le asigna un significado

Dadas las siguientes tablas y el enunciado :
Listar el codigo de cliente , nombre del cliente y el nombre de la provincia ,de aquellos clientes que se encuentran en una provincia que su nombre comience con M .

```
SELECT Clientes.codcliente,Clientes.nombre, Provincias.Descripcion
FROM Clientes
INNER JOIN Provincias ON Clientes.Cod_provincia=Provincias.Cod_Provincia
WHERE Provincias.descripcion like 'M%'
```

Parcial

Bien

El Nivel logico de abstraccion

Seleccione una:

- a. Describe como se almacenan realmente los datos
- b. Describe solo parte de la base de datos
- c. Describe que tipo de claves deben tener los usuarios
- d. Describe que datos se almacenan en la base de datos y que relaciones existen

Bien

Que valor devolverá el siguiente SQL: SELECT apellido FROM Clientes WHERE apellido LIKE '_r%'

Seleccione una:

- a. Barbieri
- b. Perez
- c. Troncoso
- d. Baru
- e. Borboton

mal

Si hablamos del concepto de Aislamiento

Seleccione una:

- a. Requiere que cada transacción sea "todo o nada": si una parte de la transacción falla, todas las operaciones de la transacción fallan, y por lo tanto la base de datos no sufre cambios
- b. Requiere que cualquier transaccion lleve a la base de datos de un estado valido a otro estado válido
- c. Asegura que una transaccion una vez confirmada (commit) quedará persistida en forma permanente.
- d. Asegura la ejecucion concurrente de las transacciones y que cada transacción debe ejecutarse en aislamiento total

bien

El lenguaje SQL esta compuesto por

Seleccione una:

- a. El lenguaje SQL, este compuesto por DML y consultas
- b. El lenguaje SQL, este compuesto por DML, DDL
- c. El lenguaje SQL, este compuesto por SQL ANSI
- d. El lenguaje SQL, este compuesto por DML, DDL y SPL

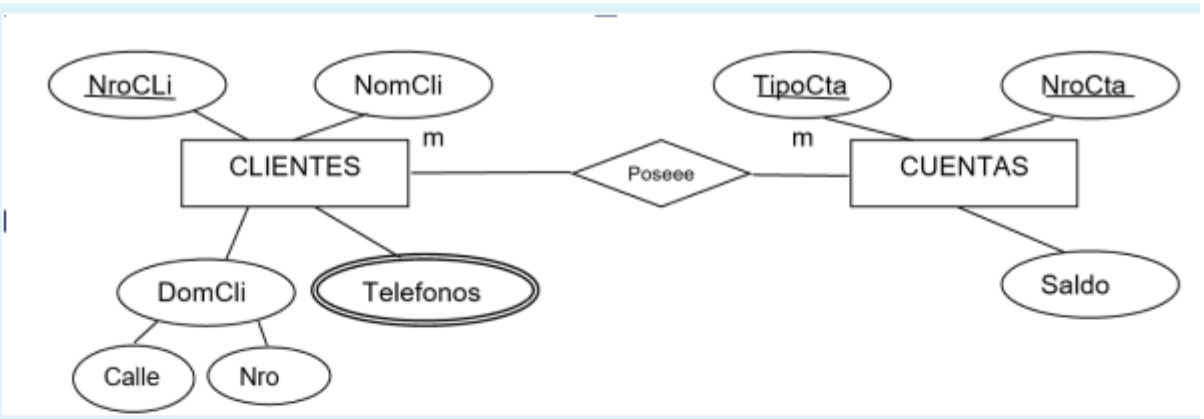
bien

Si quiero utilizar en el Where como condicion que el campo Apellido sea NULL, que sintaxis es la correcta:

Seleccione una:

- a. where apellido like 'NULL'
- b. where apellido = 'null'
- c. Where apellido is null
- d. where apellido like '%null%'
- e. Where apellido like null

mal

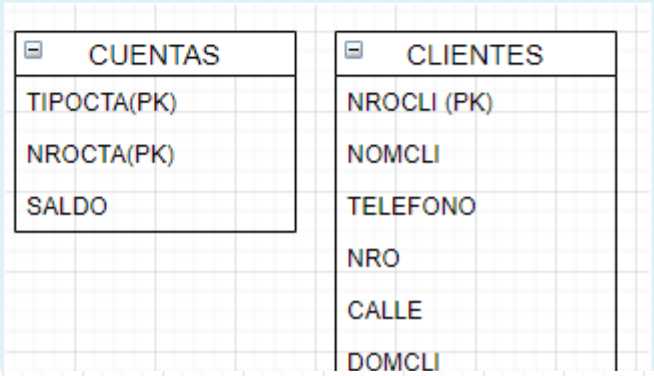


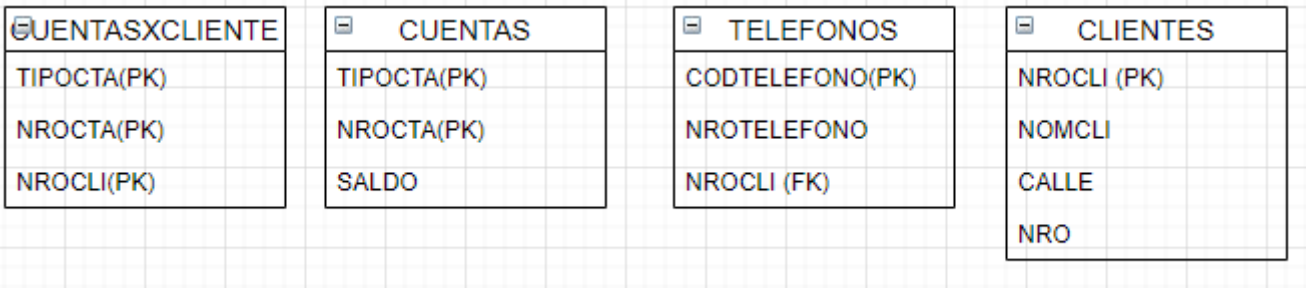
Seleccionar el DLR correcto.

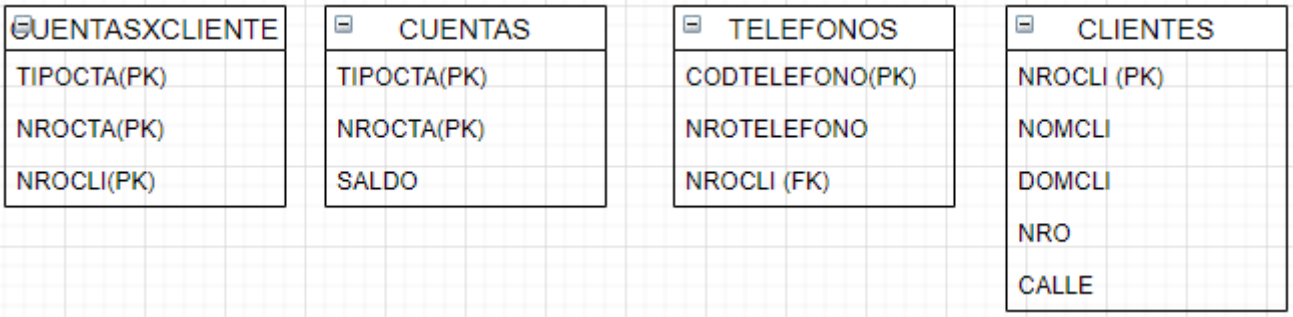
Seleccione una:

a.

b.







bien

Dadas las siguientes tablas y el enunciado:
Se desea obtener un listado de los viajes realizados a la provincias .
Mostrar en el mismo: Nombre de la provincia, Cantidad de viajes realizados, Importe Total de los viajes.

Clientes	Transporte	Viajes	Provincias
- CodCliente - Teléfono - Nombre - Apellido - Cod_Provincia - Mail	- Patente - Dni_conductor - Nombre_Conductor - Marca_auto - Modelo_auto - Cod_provincia	- Patente - CodCliente - Fecha_hora - Importe - Estado ('R' Realizado, 'C' Cancelado) - Cod_provincia	- Cod_provincia - Descripción

Seleccione la sentencia SQL correcta para resolver el mismo

Seleccione una:

a.
```
SELECT Provincias.Descripcion,
count(viajes.cod_provincia),
sum(viajes.importe)
FROM Viajes
INNER JOIN Provincias ON viajes.Cod_provincia=Provincias.Cod_Provincia
WHERE viajes.Estado="R"
GROUP BY Viajes.Cod_provincia
```

b.
```
SELECT Provincias.NombreProvincia,
count(viajes.cod_provincia),
sum(viajes.importe)
FROM Viajes
INNER JOIN Provincias ON viajes.Cod_provincia=Provincias.Cod_Provincia
WHERE viajes.Estado="R"
GROUP BY Viajes.Patente
```

c.
```
SELECT Provincias.Descripcion,
count(viajes.cod_provincia),
sum(viajes.importe)
FROM Viajes
INNER JOIN Provincias ON viajes.Cod_provincia=Provincias.Cod_Provincia
WHERE viajes.Estado="R"
ORDER BY viajes.Cod_Provincia, viajes.CodCliente
```

bien

De las siguientes consultas, marque cuál considera que es una DDL.

Seleccione una:

a. CREATE TABLE Clientes (IdClientes int, Apellido varchar(250))

b. SELECT * FROM Personas

c. UPDATE Usuarios set Email = @pEmail

bien

En el momento del pasaje de DER a DLR, cuando queremos modelar una relación de 2 entidades y su cardinalidad es de M:N... ¿qué sucede ?

Seleccione una:

- a. Se genera una nueva tabla intermedia, con las PK de ambas entidades.
- b. Se genera una tabla intermedia, sin relacion con las otras tablas
- c. Se crea una nueva entidad con la suma de los atributos de ambas entidades
- d. La entidad que tiene mas relaciones hereda la PK de la otra
- e. La entidad con mas atributos hereda la PK de la otra entidad

bien

Dadas las siguientes tablas y el enunciado :
Listar el Apellido y nombre de todos los clientes ordenado por apellido de la A a la Z .

Clientes	Transporte	Viajes	Provincias
- CodCliente - Teléfono - Nombre - Apellido - Cod_Provincia - Mail	- Patente - Dni_conductor - Nombre_Conductor - Marca_auto - Modelo_auto - Cod_provincia	- Patente - CodCliente - Fecha_hora - Importe - Estado ('R' Realizado, 'C' Cancelado) - Cod_provincia	- Cod_provincia - Descripción

Seleccione la sentencia SQL correcta para resolver el mismo

Seleccione una:

- a. Select Nombre , Apellido from Clientes order by Apellido (A-Z)
- b. Order by Apellido ASC Select Nombre , Apellido from Clientes
- c. Select Nombre , Apellido from Clientes order by Apellido Desc
- d. Select Nombre , Apellido from Clientes order by Apellido ASC
- e. Insert into Clientes (Nombre, Apellido) Values ('Carlos','Gomez')

bien

Dadas las siguientes tablas y el enunciado :
Listar todas las provincias que comiencen con M

Clientes	Transporte	Viajes	Provincias
- CodCliente - Teléfono - Nombre - Apellido - Cod_Provincia - Mail	- Patente - Dni_conductor - Nombre_Conductor - Marca_auto - Modelo_auto - Cod_provincia	- Patente - CodCliente - Fecha_hora - Importe - Estado ('R' Realizado, 'C' Cancelado) - Cod_provincia	- Cod_provincia - Descripción

Seleccione la sentencia SQL correcta para resolver el mismo

Seleccione una:

- a. Select * from Provincias Where Descripcion Like 'M%'
- b. Select * from provincias Where Cod_Provincia ='M'

mal

Dadas las siguientes tablas y el enunciado:
Se pide un listado completo de todos los clientes que hayan cancelado viajes , debe incluir Nombre, Apellido y Telefóno del cliente, Dni del conductor y Fecha.

Clientes	Transporte	Viajes	Provincias
- CodCliente - Teléfono - Nombre - Apellido - Cod_Provincia - Mail	- Patente - Dni_conductor - Nombre_Conductor - Marca_auto - Modelo_auto - Cod_provincia	- Patente - CodCliente - Fecha_hora - Importe - Estado ('R' Realizado, 'C' Cancelado) - Cod_provincia	- Cod_provincia - Descripción

Seleccione la sentencia SQL correcta para resolver el mismo

Seleccione una:

a. SELECT clientes.nombre,
clientes.Apellido,
clientes.Telefono
FROM viajes
INNER JOIN clientes ON viajes.CodCliente=Clientes.CodCliente
INNER JOIN Transporte ON viajes.Patente=Transporte.Patente
WHERE viajes.Estado="R"

b. SELECT clientes.nombre,
clientes.Apellido,
clientes.Telefono,
Transporte.Dni_conductor,
viajes.Fecha_Hora
FROM viajes
LEFT JOIN clientes ON viajes.CodCliente=Clientes.CodCliente
LEFT JOIN Transporte ON viajes.Patente=Transporte.Patente

Esta es

e. SELECT clientes.nombre,
clientes.Apellido,
clientes.Telefono,
Transporte.Dni_conductor,
viajes.Fecha_Hora
FROM clientes
INNER JOIN Transporte ON Clientes.CodCliente=Transporte.CodCliente and
viajes.Patente=Transporte.Patente
WHERE viajes.Estado="C"

bien

¿Cuál de las siguientes consultas es correcta?

Seleccione una:

a. SELECT P.DNI, P.Nombre, C.Valor FROM Personas P LIMIT JOIN Compras C ON P.DNI = P.IDCliente

b. SELECT P.DNI, P.Nombre, C.Valor FROM Personas P INNER JOIN Compras C GROUP BY P.Nombre

c. SELECT P.DNI, P.Nombre, C.Valor FROM Personas P INNER JOIN Compras

d. SELECT P.DNI, P.Nombre, C.Valor FROM Personas P INNER JOIN Compras C ON P.DNI = C.IDCliente

e. SELECT DNI.P, Nombre.P, Valor.P FROM Personas P INNER JOIN Compras C IN P.DNI = C.IDCliente

mal

Si queremos seleccionar todos los datos de los clientes que sean de la localidad de claromeco y su apellido comienza con W , debo utilizar la siguiente consulta :

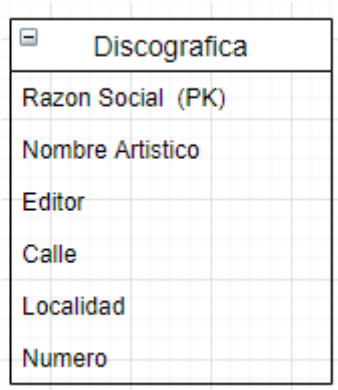
Seleccione una:

- a. Select * from clientes C inner join Localidad L ON C.localidadId=L.localidadId where l.nombrelocalidad = 'claromeco' and C.apellido = 'W%'
- b. Select Clientes.* from clientes Localidad L ON C.localidadId=L.localidadId where l.nombrelocalidad like 'claromeco' and C.apellido like 'W%'
- c. Select Clientes.* from clientes C inner join Localidad L ON C.localidadId=L.localidadId where l.nombrelocalidad like 'claromeco' and C.apellido like '_W_'
- d. Select Clientes.* from clientes C inner join Localidad L ON C.localidadId=L.localidadId where l.nombrelocalidad = 'claromeco' and C.apellido like '%W'
- e. Select Clientes.* from clientes C inner join Localidad L ON C.localidadId=L.localidadId where l.nombrelocalidad = 'claromeco' and C.apellido like 'W%'

bien

Seleccionar el DLR que sea correcto según el siguiente DER.

d.



bien

Los atributos multivalorados son aquellos que

Seleccione una:

- a. pueden tener varios valores posibles
- b. tienen subatributos
- c. son minimos y son indivisibles
- d. derivan de otros atributos

bien

Los atributos simples

Seleccione una:

- a. derivan de otros atributos
- b. pueden tener varios valores posibles
- c. son minimos y son indivisibles

bien

La defincion de DATO es

Seleccione una:

- a. Representación de hechos, conceptos, o instrucciones, a la que se le asigna un significado
- b. Representacion de informacion
- c. Un valor en una tabla

mal

Si hablamos del concepto de atomicidad

Seleccione una:

- a. Requiere que cada transacción sea "todo o nada": si una parte de la transacción falla, todas las operaciones de la transacción fallan, y por lo tanto la base de datos no sufre cambios
- b. Requiere que cualquier transaccion lleve a la base de datos de un estado válido a otro estado válido
- c. Asegura la ejecución concurrente de las transacciones.
- d. Una transacción una vez confirmada (commit) quedará persistida en forma permanente.

bien

Los modelos de datos son

Seleccione una:

- a. una colección de herramientas conceptuales para describir los datos,las relaciones, la semántica y las restricciones de consistencia.
- b. una colección de herramientas para poder representar las relaciones entre varias tablas.
- c. son una representacion en una base de datos de un modelo conceptual

bien

Si hablamos de una colección de operaciones que se lleva a cabo como una única función lógica en una aplicación de bases de datos

Seleccione una:

- a. hablamos de una clave primaria
- b. hablamos de un índice
- c. hablamos de un backup
- d. hablamos de una transacción
- e. hablamos de una consulta SQL
- f. hablamos de un restore

Pregunta **15**

Finalizado

Puntúa 1,0 sobre 1,0

Marcar pregunta

La claúsula ORDER BY dentro de una sentencia SQL...

Seleccione una:

- a. Actualiza los registros en la base de datos, alterando el orden de los mismos
- b. Agrupa los resultados por las columnas seleccionadas en la consulta
- c. Realiza un filtro sobre las columnas indicadas, aplicando una búsqueda parcial sobre las mismas
- d. Muestra los resultados ordenados de acuerdo a la/las columnas indicadas
- e. Realiza una conexión entre claves PK y FK de las tablas indicadas

https://www.w3schools.com/sql/sql_syntax.asp

https://drive.google.com/drive/folders/1VaXrPFiiLW_jlYVRc0YSWru0VkPTRHxy

---

EXTRA

SQL (Lenguaje de consultas estructurado)

SQL (Lenguaje de consultas estructurado)

• En el motor de base de datos vamos a usar el lenguaje Transact-Sql.

• Para las consultas y modificación de datos se utiliza el lenguaje ANSI SQL que viene a ser el primer lenguaje de bases de datos relacionales.

• Se agregan funciones básicas para datos como fechas, strings etc.

• Gracias a las consultas SQL, podremos administrar los datos en nuestra base, agregar, modificar, eliminar, consultar, y otras operaciones.

Lenguaje de definición de datos (DDL)

• Las sentencias DDL se utilizan para crear y modificar la estructura de las tablas así como otros objetos de la base de datos.

• CREATE - para crear objetos en la base de datos.

• ALTER - modifica la estructura de la base de datos.

• DROP - borra objetos de la base de datos.

• TRUNCATE - elimina todos los registros de la tabla, incluyendo todos los espacios asignados a los registros. Además, reinicia los campos autonuméricos.

Lenguaje de manipulación de datos (DML)

• Las sentencias de lenguaje de manipulación de datos (DML) son utilizadas para gestionar datos dentro de las bases de datos . Algunos ejemplos:

• SELECT - para obtener datos de una base de datos.

• INSERT - para insertar datos a una tabla.

• UPDATE - para modificar datos existentes dentro de una tabla.

• DELETE - elimina todos los registros de la tabla; no borra los espacios asignados a los registros.