

BASE DE DATOS SQL

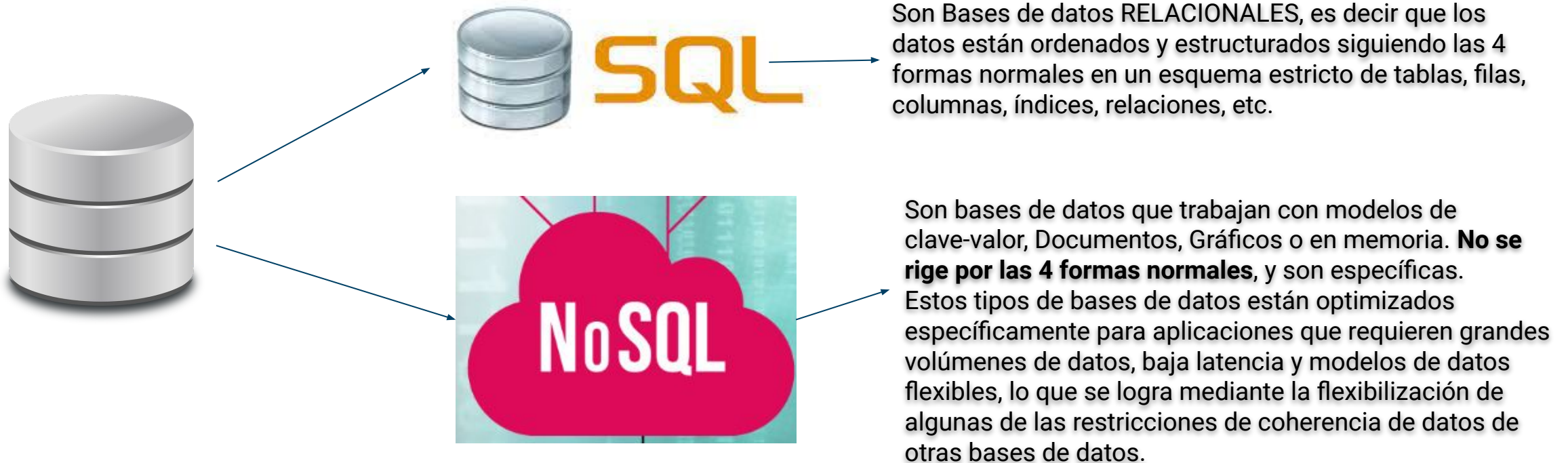


RESUMEN SQL

BASE DE DATOS

Una **base de datos** es una recopilación organizada de información o **datos** estructurados, **que** normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una **base de datos** está controlada por un sistema de gestión de **bases de datos** (DBMS) o MOTOR DE BASE DE DATOS. Entre estos los más conocidos son PL/SQL, SQL SERVER, MY SQL, ETC.

Existen varios tipos de DDBB (Bata Base) según su su estructura, contexto, utilidad y necesidades. En este curso vamos a mencionar a 2 principales que se diferencian principalmente en su lógica de estructura.



BASE DE DATOS RELACIONALES

NORMALIZACION

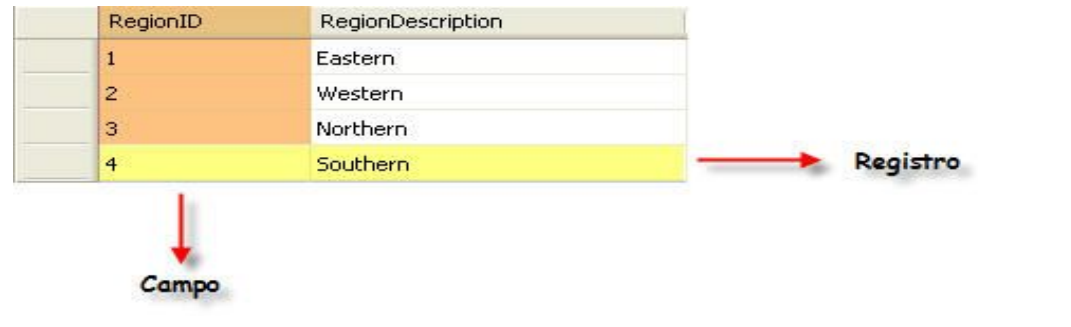
La **normalización** es un concepto de diseño de bases de datos que se aplica a las bases de datos relacionales **para evitar las redundancias**.

El modelo relacional es el concepto más extendido en la gestión informatizada de los datos. En las bases de datos de este tipo, **la información se guarda en registros en tablas interconectadas por medio de claves**. Un registro se compone de varios campos de valores que se subordinan a ciertos atributos a lo largo de las columnas de la tabla.

TABLAS / ENTIDADES

En las tablas vamos a ordenar los datos y su estructura se basa en **REGISTROS** (filas, cdr, etc) y **CAMPOS** (columnas, Atributos, etc)

RegionID	RegionDescription
1	Eastern
2	Western
3	Northern
4	Southern



REGISTROS: Es el conjunto de campos, ordenados que se grabarán en tabla. Cada fila de la tabla es 1 registro.

- **SON LOS DATOS.**
- **SE PUEDEN MODIFICAR**
- **NO ALTERA LA ESTRUCTURA**
- **DEPENDE DE LOS CAMPOS**

BASE DE DATOS RELACIONALES

CAMPOS: cada dato pertenece a un CAMPO.

Eje. si decido guardar el NOMBRE y el APELLIDO podría guardarlo en 1 campo (nombre_completo) o 2(nombre ; apellido).

Tener en cuenta que al juntar los 2 datos al momento de realizar una búsqueda como resultado obtendré la información almacenada en el mismo. Si yo guarde el nombre y el apellido todo en 1 solo campo **NO PODRÉ SEPARAR EL NOMBRE DEL APELLIDO.**

La cantidad de columnas es el GRADO de la tabla.

Debe tener un **NOMBRE** (normalmente son en minúscula y los espacios son con guión bajo).

Tienen un **TIPO DE DATO** -> el tipo de dato es una restricción al campo. Ejm. el DNI es un dato **INT** (Entero), por ende ese campo solo acepta números.

También se determina la longitud del campo por ejm. **int (8)** es decir que el entero que se guardará en ese campo no puede tener más de 8 caracteres.

TIPOS DE DATOS MÁS UTILIZADOS

- VARCHAR (String/texto) -> nombre VARCHAR(100)
- INT (entero) -> dni INT (8)
- FLOAT/DECIMAL -> precio FLOAT(12,2), precio2 DECIMAL(12,2)
- DATE -> fecha DATE

[Pueden ver la documentación de MySQL](#)

IMPORTANTES

- HACEN A LA ESTRUCTURA.
- NO ES RECOMENDABLE REALIZAR CAMBIOS
- AFECTA A LA ESTRUCTURA DE DATOS.
- PUEDEN SER ÍNDICES DE LA TABLA.
- SE DEFINEN LAS KEY (Claves)

CLAVES

PRIMARIA (PK) Y CLAVE FORÁNEA (FK)

Las KEY o Claves son CAMPOS que tienen las tablas para relacionarse entre sí, y además controlan la integridad de los datos.

Estas Key son definidas por el Administrador de base de datos y es muy sencillo de comprender con un ejemplo:

- Se creó una tabla que registra los clientes de un negocio, el problema es que los usuarios dicen que les cuesta mucho tiempo buscar al cliente. Al ver la tabla se ve algo parecido a lo siguiente:

nombre	apellido	dni
juan	palotes	123123123
pepe	perengano	456456456
lucio	ramos	456456564

La demora que mencionan los usuarios radica en la tabla que carece de índice. Al realizar una búsqueda solo por el nombre el motor de base de datos busca caracter por caracter su equivalencia en el campo de búsqueda, en caso de DB pequeñas la demora se puede manejar pero en bases muy grandes puede tildar el motor o no permitir el acceso a otros usuarios

La solución a esa demora es crear un campo PK (Primary Key) o Clave Primaria.

- Suele ser el primer campo de la tabla.
- Lleva el prefijo ID.
- Se completa con información a que tabla pertenece **id_clientes**
- El id es único e irrepetible.**
- Puede ser incremental o no.**
- Puede definirse más de 1 campo. ejm. id_cliente, id_cuenta (lo veremos mas adelante).**
- NO PUEDE SER NULL.**
- NOS PERMITE ENLAZAR (JOIN) CON OTRAS TABLAS**

TABLA 1			
id	nombre	apellido	dni
1	juan	palotes	123123123
2	pepe	perengano	456456456
3	lucio	ramos	456456564

AHORA EN VEZ DE BUSCAR POR EL NOMBRE
BUSCA POR ID-> SU PK

Es decir que con la FK le digo a la tabla que ese campo es PK de otra.

Al querer eliminar el domicilio, el Motor nos alerta que ese registro está vinculado con otro registro, de otra tabla (FK), en caso de eliminar el domicilio(PK) debe eliminar los registros donde están vinculados FK (CONSISTENCIA DE LA BASE). Controla que las tablas vinculadas al verse afectadas por la PK no genere inconsistencias

(en estos casos los registros tienen un campo **estado** que es de uso interno **que no elimina el registro de la DB** pero según ese estado puede estar activo o inactivo)

[illegible]

A TRABAJAR CON LÁPIZ Y PAPEL

Vamos a trabajar en la normalización, para evitar **REDUNDANCIA** sino también **INCONSISTENCIAS** en los datos.
Es información inexacta, elementos que no coinciden entre sí.

ENTENDIENDO COMO ES LA LÓGICA DE LA PK y FK

QUE PASA SI EN MI PLANILLA DE EXCEL QUIERO FILTRAR
POR CÓRDOBA?

- Como Normalizarían el campo CIUDAD y PROVINCIA?

TABLA 2						
id_domicilio(PK)	calle	numero	barrio	cp	ciudad	provincia
1	Av. Landa	155	Alberdi	5000	CORDOBA	CORDOBA
2	Av Siempre Vivas	1234	Gnral PAz	500	cor	cba
3	Neuquen	368			CBA	CBA
4	Chaco	456	centro		CORDOBA	CORDOBA

Se puede ver que el usuario a ingresado texto en los campos CIUDAD Y PROVINCIA pero que son inconsistentes. Los HUMANOS entendemos que CORDOBA = CBA = cba = CorDoBa, etc pero el motor solo lee caracteres y CORDOBA != CBA && CORDOBA != CorDoBa

RESPUESTA Y TÉCNICA

Para normalizar la tabla de **DOMICILIOS** debemos crear una tabla **CIUDADES** que va a tener una **PK** que va a ser la **FK** de la tabla domicilios y quedaria asi:

TABLA CIUDADES		
id_ciudad	descripcion	id_provincia
1	CIUDAD DE CORDOBA	1
2	CABA	2
3	SALVADOR	

TABLA PROVINCIAS		
id_provincia	descripcion	id_pais
1	CORDOBA	1
2	BUENOS AIRES	1
3	BAHIA	3

TABLA PAISES	
id_pais	descripcion
1	ARGENTINA
2	BRASIL
3	URUGUAY

TABLA 2					
id_domicilio	calle	numero	barrio	cp	id_ciudad
1	Av. Landa	155	Alberdi	5000	1
2	Av Siempre Vivas	1234	Gnral PAz	500	1
3	Neuquen	368			1
4	Chaco	456	centro		1

La misma lógica aplicamos para las **PROVINCIAS** y para **PAÍSES**

El tener normalizados los datos nos permite realizar nuevas combinaciones de datos, por ejemplo: **Quisiera registrar la nacionalidad del cliente independientemente de su domicilio actual.**

Simplemente a la tabla **clientes** le agregamos el campo **id_nacionalidad(FK)** y ahora podemos usar la tabla **PAÍSES** como **NACIONALIDAD**

LAS BÚSQUEDAS REALIZADAS POR PK y FK SON MUY VELOCES EN COMPARACIÓN DE OTROS CAMPOS, YA QUE ESTÁN INDEXADAS.

A TRABAJAR CON SQL

¿QUE ES SQL?

SQL (por sus siglas en **inglés** **Structured Query Language**; en **español** **lenguaje de consulta estructurada**) es un **lenguaje de dominio específico**, diseñado para administrar, y recuperar información de **sistemas de gestión de bases de datos relacionales**.² Una de sus principales características es el manejo del **álgebra** y el **cálculo relacional** para efectuar consultas con el fin de recuperar, de forma sencilla, **información** de **bases de datos**, así como realizar cambios en ellas.

¿QUÉ TIPO DE CONSULTAS/QUERY EXISTEN?

Existen 2 tipos de consultas en el lenguaje SQL y que son muy fáciles de distinguir.

ESTRUCTURALES

- Se utilizan para manipular la estructura de la DB y las tablas. (aplica a los campos y no a los registros)
- Las 3 principales son:
 - CREATE -> sentencia para crear. [Link](#) [Link2](#)
 - DROP -> sentencia para eliminar. [Link](#) [Link2](#)
 - ALTER -> sentencia para modificar. [Link](#)

NO ESTRUCTURALES

- Se utilizan para manipular los datos de la DB. Mostrar, agregar, modificar o eliminar registros o datos de las tablas.
- Las 4 principales son:
 - SELECT -> sentencia obtener datos. [Link](#)
 - INSERT-> sentencia para insertar datos. [Link](#)
 - UPDATE -> sentencia para modificar. [Link](#)
 - DELETE -> sentencia para eliminar. [Link](#)

MANOS A LA OBRA

TAREA:

- 1 - Crear una Base de datos a la que llamaremos "TEST"
- 2 - Crear las tablas Clientes, Domicilios, Ciudades, Provincias, Paises
- 3 - Crear las PK y FK de cada tabla.
- 4 - Insertar 5 registros en cada tabla.
- 5 - Modificar 3 registros de cada tabla.
- 6 - Eliminar 1 registro de cada tabla
- 7 - Crear una sentencia SELECT que liste a los clientes.
- 8 - Crear una sentencia SELECT con una condición (WHERE). [link](#)
- 9 - Crear una sentencia SELECT donde el FROM son 2 tablas relacionadas (JOIN). [link](#)
- 10 - Crear una sentencia SELECT donde no se vean los id de las tablas y solo se lea las descripciones.

Todas las consultas deberán estar guardadas en un archivo con el nombre. TP1_CONSULTAS.SQL

AHORA QUE SABEMOS RELACIONAR TABLAS

- ¿Qué criterios debemos tener para relacionar entidades?
- ¿Cómo creamos las tablas Tablas?

Las formas normales son una serie de mecanismos que proporcionan criterios para detectar anomalías e inconsistencias en nuestras tablas. Existen desde la 1FN hasta la 6FN. Cuanto más alta sea la forma normal aplicable a una tabla, menos vulnerable será a inconsistencias y anomalías. La numeración de las formas normales no significa que sea un proceso restrictivo por el cual haya que pasar sí o sí. Con la práctica, es probable que nuestros diseños de bases de datos estén al menos en la 3FN y no haya hecho falta normalizar desde 1FN hasta 2FN y después 3FN.

PRIMERA FORMA NORMAL

La 1FN es la forma mínima en la que se encuentran nuestras tablas. Consiste en añadir una tabla adicional para satisfacer los criterios mínimos de integridad, consistencia y concurrencia. Esta forma normal se utiliza para representar una relación básica y evitar grupos repetitivos. Una tabla está en 1FN si se cumplen las siguientes condiciones:

- Todos los datos son **atómicos**.
- Todas las columnas contienen el mismo tipo de datos.

Un registro se considera atómico cuando a cada información (cada asunto) se le reserva una celda propia.

¿Cómo Cumplir con la 1FN? -> como ya hemos hecho con el ejemplo de la tabla clientes y domicilios

1. Subdividir todos los datos multivalor en columnas separadas.
2. Comprobar que los valores en cada columna son del mismo tipo.

SEGUNDA FORMA NORMAL

SEGUNDA FORMA NORMAL

Para estar en la segunda forma normal, a las condiciones de la primera se añade la siguiente:

- Los atributos que no forman parte de ninguna clave han de depender funcionalmente de toda la clave primaria. [Link](#)

INTENTANDO SER MÁS CLARO

Si existe una CLAVE PRIMARIA (PK) el resto de los datos que no son FK dependen directamente de la clave primaria.

cuando no se genera esa dependencia es por que NO CUMPLE con 2FN.

NO CUMPLE

TABLA 1			
id_cliente (PK)	nombre	apellido	dni
1	juan	palotes	123123123
2	pepe	perengano	456456456
3	juan	palotes	123123123

No cumple con la 2FN porque para el cliente con el DNI 123123123 existen 2 claves primarias. O también para JUAN PALOTES existen 2 registros.

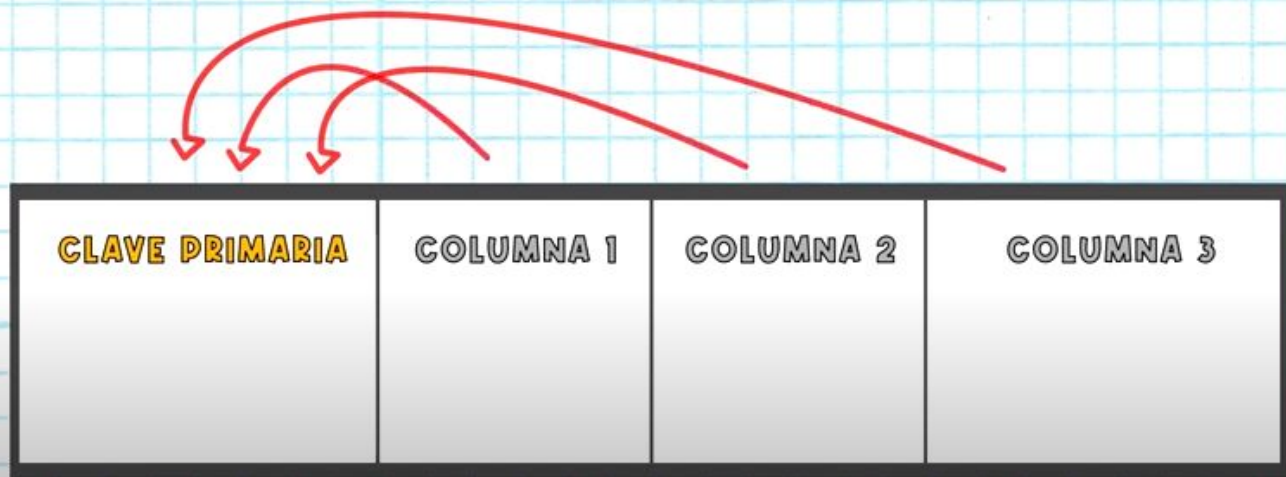
SI CUMPLE

TABLA 1			
id_cliente (PK)	nombre	apellido	dni
1	juan	palotes	123123123
2	pepe	perengano	456456456
3	marina	sarabia	123123124

Cumple con la 2FN porque para el cliente con el dni 123123124 existe 1 sola clave primaria. O no existen dependencias externas.

DEPENDENCIAS FUNCIONALES Y TRANSITIVAS

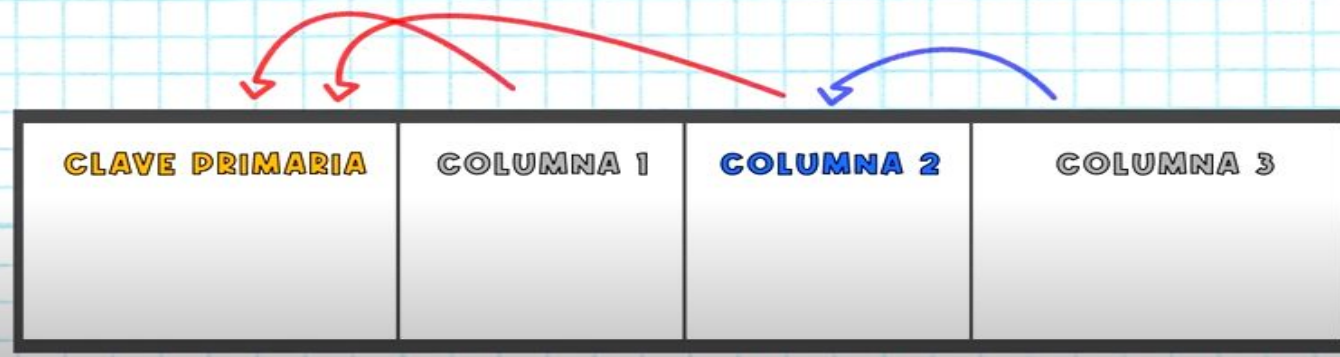
DEPENDENCIAS FUNCIONALES



Todos los atributos, que no son Key, dependen de su clave primaria (PK).

Cuando uno de los atributos NO depende de la clave primaria. PERO su clave primaria ESTA en la misma tabla y esta SI depende de la PK.

DEPENDENCIAS TRANSITIVAS



TERCERA FORMA NORMAL

Para cumplir con la 3FN las entidades deben cumplir con :

- Que se estén cumpliendo las 1FN y 2FN.
- NO tener dependencias transitivas.

Como vimos antes una dependencia transitiva es cuando dentro de la tabla tenemos una PK que no pertenece a la tabla pero tampoco juega como FK. un ejemplo sería el siguiente:

id_domicilio (PK)	calle	numero	barrio	cp	id_ciudad	ciudad
1	Av. Landa	155	Alberdi	5000	1	CIUDAD DE CORDOBA
2	Av. Siempre	1234	General Paz	5009	1	CIUDAD DE CORDOBA
3	Neuquen	368			2	CABA
4	chaco	456	Centro		3	SALVADOR

Vemos que el campo **CIUDAD** tiene una dependencia directa con el **ID_CIUDAD** que esta es tiene una dependencia directa con la PK de la tabla domicilio, pero **CIUDAD** no depende de la PK **id_domicilio** sino de la PK **id_ciudad**.

TERCERA FORMA NORMAL

LA INCONSISTENCIA RADICA EN QUE HAY UN CAMPO DE MÁS.

Si el id_ciudad es = 1 a que ciudad corresponde? CIUDAD DE CORDOBA o CABA

Si la CIUDAD es CABA a que id_ciudad corresponde? 1 o 2 ?

id_domicilio (PK)	calle	numero	barrio	cp	id_ciudad	ciudad
1	Av. Landa	155	Alberdi	5000	1	CIUDAD DE CORDOBA
2	Av. Siempre	1234	General Paz	5009	1	CABA
3	Neuquen	368			2	CABA
4	chaco	456	Centro		3	SALVADOR

¿Cómo harían para que la tabla cumpla con la TERCERA FORMA NORMAL?

Existen otras reglas de normalización que se complementan a las aprendidas pero en este curso llegaremos a ver las 3 primeras ya que son suficientes para las tareas que vamos a realizar como fullstak, pero es una buena práctica revisarlas por completo a la hora de diseñar una base de datos.

Recuerden que no siempre se podrá cumplir con todas las formas normales pero si deberían cumplirse con las 3 primeras que son fundamentales para la estructuración y normalización de los datos.

También podemos ver que con la “TÉCNICA” aprendida en la cursada las formas normales se cumplen casi sin forzarlas.

Recuerden siempre focalizar en

NO SER REDUNDANTES -> repetir datos

SER CONSISTENTES -> un dato es el mismo en toda la base (el id_cliente = 1 es el mismo cliente en toda la base)