



**Argentina  
programa  
4.0**



Ministerio de Economía  
**Argentina**

Secretaría de  
Economía del Conocimiento

***primero  
la gente***

# Clase 22: Bases de datos Relacionales

## El modelo relacional

## Agenda de hoy

- A. Tipos de índices en SQL
  - a. Claves primarias, foráneas, concatenadas y candidatas
- B. Formas normales
  - a. cuántas son
  - b. primera forma normal
  - c. segunda forma normal
  - d. tercera forma normal
  - e. forma normal de (*Boyce-Codd*)
- C. Subconsultas SQL
- D. Select Case - When



# El modelo relacional

Las tablas relacionales son un punto clave y crítico en el universo de las bases de datos SQL.

Es un tema que debemos aprender a dominar, porque el modelo relacional es el Core de este tipo de base de datos, y para lograr esto, debemos entender cuántos y qué tipos de índices podemos manejar con SQL.



# El modelo relacional

A continuación haremos un repaso de los diferentes índices que pueden implementarse con el lenguaje SQL, para luego entrar en un espacio teórico donde hablaremos de las diferentes **Formas Normales**, para entender qué son y cómo sacar provecho de las mismas.

*¡Vayamos entonces a la acción!*



# Tipos de índices

# Tipos de índices

**Los índices en bases de datos relacionales son estructuras de datos utilizadas para mejorar el rendimiento de las consultas.**

Éstos funcionan como un catálogo que permite a la base de datos encontrar rápidamente las filas que satisfacen una determinada consulta.

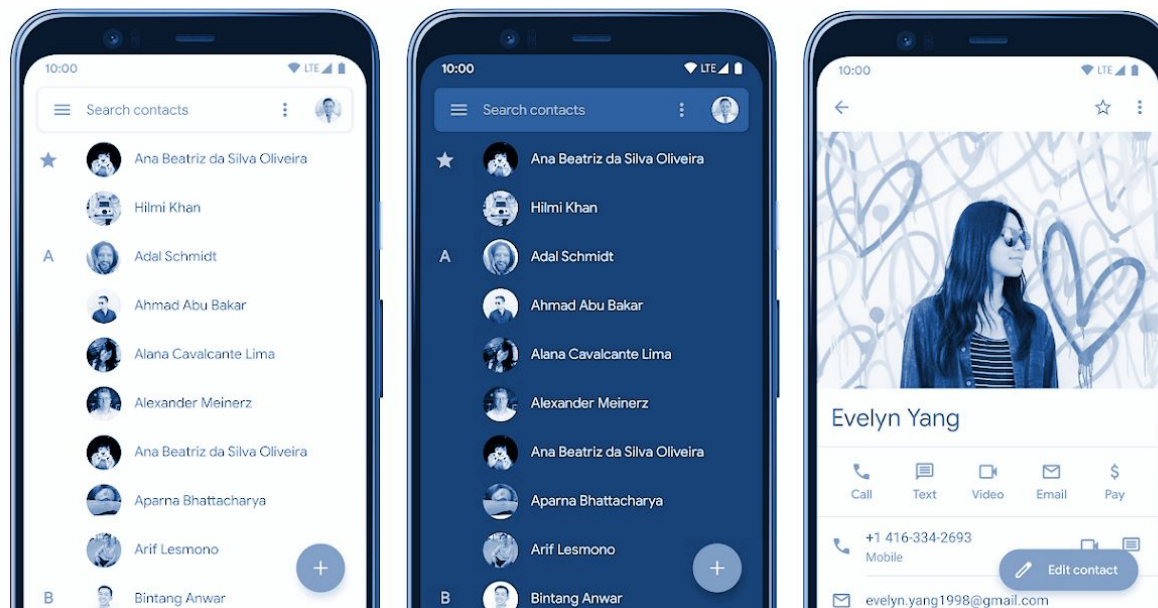


## Tipos de índices

**Para poder entender la finalidad de un índice, pensemos en la agenda de contactos en nuestro smartphone.**

Esta agenda asigna un ordenamiento alfabético a los contactos almacenados en la misma.

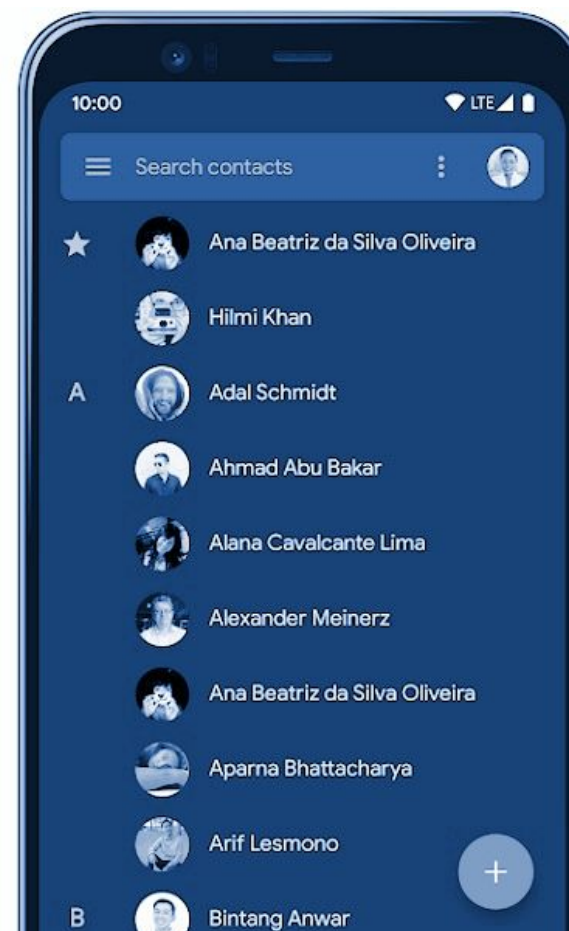
Al igual que en una agenda, SQL recomienda siempre dar un índice, numérico en este caso, a cada uno de los registros que almacenemos en una tabla.





## Tipos de índices

Y de la misma forma en la cual la agenda de contactos nos permite ubicar más rápido una persona para llamarla o enviarle un mensaje de texto, las bb.dd SQL toman esta filosofía para crear índices que permitan encontrar más rápidamente uno o más registros en una consulta de selección.



## Tipos de índices

Por ello, cuando creamos una tabla, siempre es necesario crear un campo índice como primer campo de la tabla en cuestión.

Cuando se crea un índice en una tabla, la bb.dd crea una copia ordenada de las columnas del índice, lo que le permite buscar más rápidamente los valores de las filas que cumplen con los criterios de búsqueda, acelerando las consultas que buscan en grandes conjuntos de datos.

1 • `SELECT * FROM Northwind.Employees;`

100% 1:1

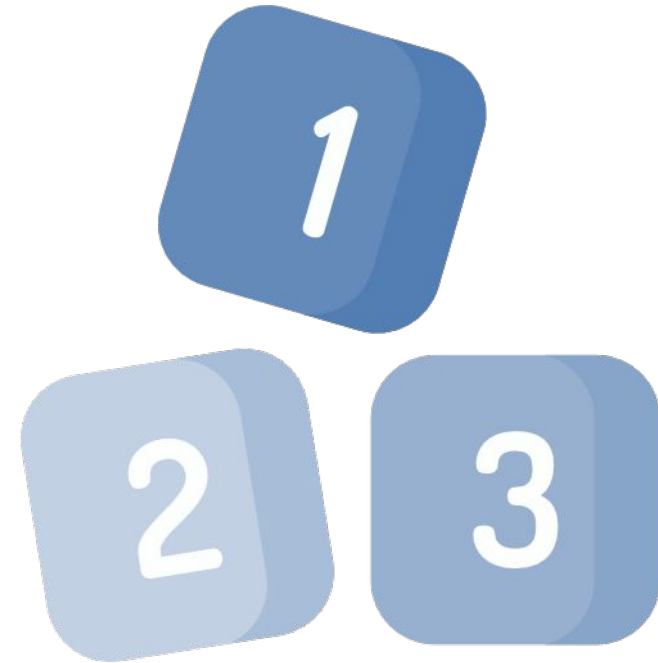
Result Grid Filter Rows: Search Edit: Export/Import:

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy
1	Davolio	Nancy	Sales Representative	Ms.
2	Fuller	Andrew	Vice President, Sales	Dr.
3	Leverling	Janet	Sales Representative	Ms.
4	Peacock	Margaret	Sales Representative	Mrs.
5	Buchanan	Steven	Sales Manager	Mr.
6	Suyama	Michael	Sales Representative	Mr.
7	King	Robert	Sales Representative	Mr.
8	Callahan	Laura	Inside Sales Coordinator	Ms.
9	Dodsworth	Anne	Sales Representative	Ms.

# Tipos de índices

Los índices se pueden aplicar a una o varias columnas en una tabla, y hay varios tipos de índices que se pueden utilizar dependiendo del tipo de datos y las operaciones que se realizan en la tabla. Entre los diferentes índices que podemos definir en las tablas relacionales, encontramos los siguientes tipos:

- Primary Key
- Foreign Key
- Candidate Key
- Concatenated Key



Primary key

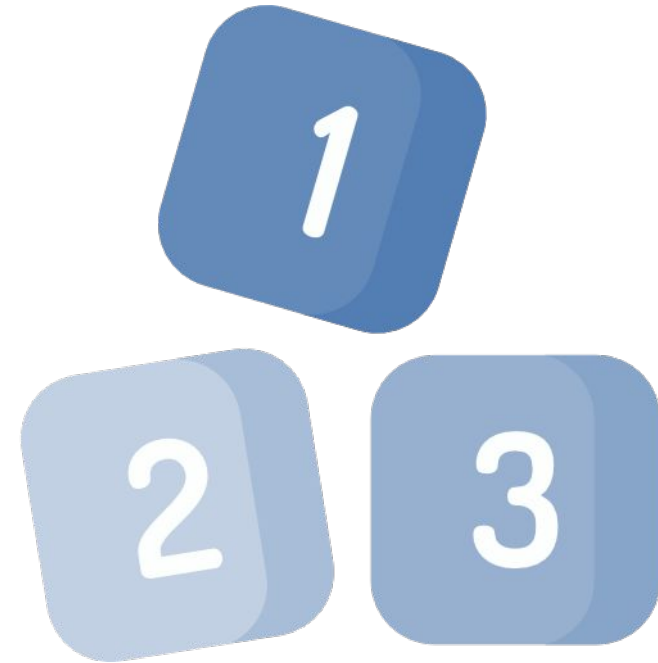
# Tipos de índices

## Primary Key

También llamado llave o clave primaria, hace que el registro sea unívoco y, obligatoriamente, no nulo.

**PK = primary key** (*es la forma de abreviarla que se aplica en las definiciones de claves al momento de diseñar una tabla*).

Una clave primaria comprende de esta manera una columna o conjunto de columnas. No puede haber dos filas en una tabla que tengan la misma clave primaria.



# Foreign key

# Tipos de índices

## Foreign Key

También llamada clave foránea, clave secundaria, o clave externa, puede ser -o no- una clave primaria dentro de una tabla.

Su característica convierte a este tipo de clave en el punto de enlace con otra tabla donde ésta (*foreign key*) tiene el rol de *primary key*.

**FK = foreign key**



## Ejemplo de Primary & Foreign keys



## Tipos de índices

	PK	Contactos					FK
	contactoID	nombre	apellido	domicilio	telefono	email	IDTipoTel
1	1	Fernando	Moon	Tandil 1234	1122223333	fernando.luna@istea.com.ar	5
2	2	Omar	Mercado	Tandil 1235	1122223444	omar.mercado@istea.com.ar	2
3	3	Pepe	Argento	Bonifacio 1234	111122224444	pepe.argento@quefamilia.com	NULL
4	4	Moni	Argento	Bonifacio 1234	1122229999	moni.argento@quefamilia.com	4
5	5	Julian	Moon	Tandil 1234	1199997766	juli.moon@gmail.com	5
6	6	Joe	McMillian	Texas 123	14455667788	jmm@cardiffcomputers.com	NULL

Veamos un ejemplo de una tabla de contactos, similar a la que creamos en nuestras primeras clases de SQL. Aquí podemos apreciar que, las claves foráneas, a diferencia de las claves primarias, repetirán varias veces un valor en la tabla donde ofician como tales.

Solo hacen de nexo con otra tabla.

## Tipos de índices

**PK**

	IDTipoTel	Descripcion
1	1	N/A
2	2	Particular
3	3	Trabajo
4	4	Móvil
5	5	Otro
6	6	Skype
7	7	Satélital

**TiposDeTelefono**

**PK**

**Contactos**

**FK**

	contactoID	nombre	apellido	domicilio	telefono	email	IDTipoTel
1	1	Fernando	Moon	Tandil 1234	1122223333	fernando.luna@istea.com.ar	5
2	2	Omar	Mercado	Tandil 1235	1122223444	omar.mercado@istea.com.ar	2
3	3	Pepe	Argento	Bonifacio 1234	111122224444	pepe.argento@quefamilia.com	NULL
4	4	Moni	Argento	Bonifacio 1234	1122229999	moni.argento@quefamilia.com	4
5	5	Julian	Moon	Tandil 1234	1199997766	juli.moon@gmail.com	5
6	6	Joe	McMillian	Texas 123	14455667788	jmm@cardiffcomputers.com	NULL

Las claves primarias (**PK**) de aquellas tablas que se interpretan como secundarias, son las claves foráneas (**FK**) de tablas con mayor protagonismo.

# Concatenated key

# Tipos de índices

## Concatenated Key

Es un tipo de clave que ayuda a encontrar la singularidad en una tabla combinando dos o más campos.

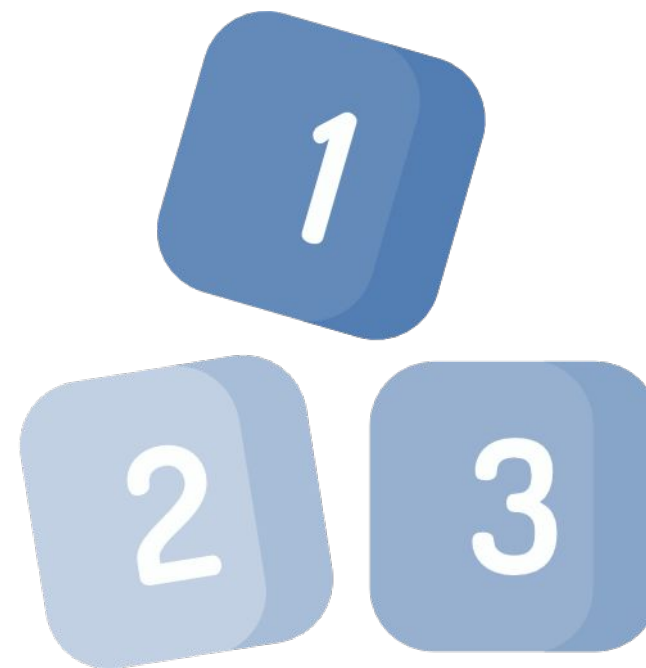
Las claves concatenadas suelen darse en determinadas situaciones donde dos o más campos deben combinarse. Uno de los ejemplos que podemos tomar como referencia es una tabla que almacene información del encabezado de una factura de compra.



# Tipos de índices

## Concatenated Key

Aunque en muchos casos puede haber en una tabla una clave primaria que gestione esto, las claves concatenadas pueden llegar a ser una alternativa para prevenir posibles otras fallas en un diseño que terminen arrastrando problemas hacia otros procesos laborales.



## Tipos de índices

PK	CK				encabezadoFactura					
IDFac	TipoFactura	Letra	PuntoVenta	NumeroFactura	IDCliente	FechaFactura	FechaVencimiento	NotaCredito	NotaDebito	Cancelada
21375	FC	C	2	1234	123	6/7/2022	6/9/2022	0	0	False
21376	FC	C	2	1235	202	6/7/2022	6/9/2022	0	0	False
21377	FC	C	2	1236	229	6/7/2022	6/9/2022	0	0	False
21378	FC	C	2	1237	213	6/7/2022	6/9/2022	0	0	False
21379	FC	C	2	1238	147	7/7/2022	7/9/2022	0	0	False
21380	FC	C	2	1239	308	7/7/2022	7/9/2022	0	0	False
21381	FC	C	2	1240	211	7/7/2022	7/9/2022	0	0	False
21382	FC	C	2	1241	607	7/7/2022	7/9/2022	0	0	False
21383	FC	C	2	1242	217	7/7/2022	7/9/2022	0	0	False
21384	FC	C	2	1243	1	8/7/2022	8/7/2022	0	0	False

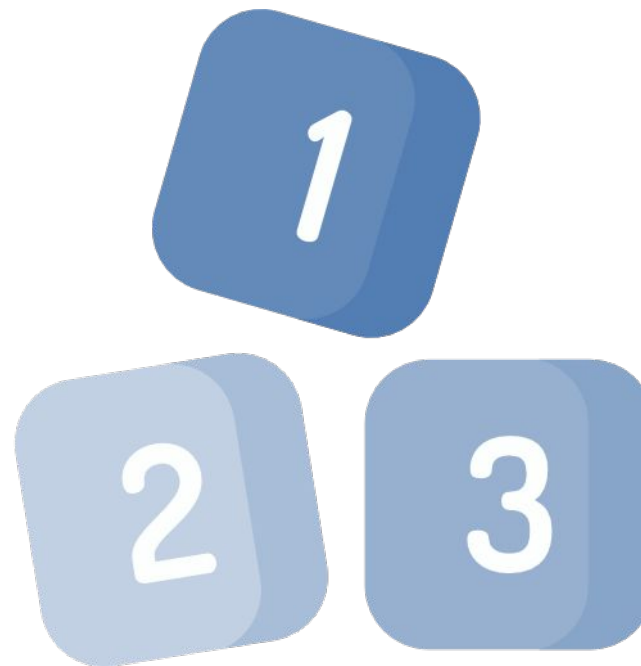
Implementación de una clave concatenada sobre una tabla de facturación. La combinación de los cuatro campos definidos después del índice primario, impedirá la duplicación de una factura teniendo en cuenta la combinación de todos ellos.

# Candidate key

# Tipos de índices

## Candidate Key

Las claves candidatas suelen ser una clave primaria con una estructura numérica o alfanumérica, que usualmente ayudan a realizar búsquedas por algún valor más conocido que el que puede aportar una clave primaria convencional.



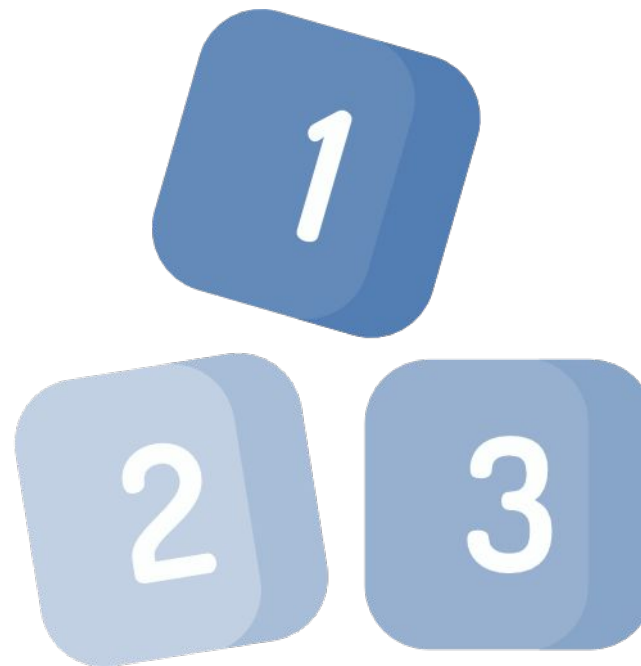


## Tipos de índices

### Candidate Key

Para entender esto, pensemos en una bb.dd. que gestiona los empleados de una empresa. En la tabla donde se almacenan estos empleados, seguramente existirá un campo llamado **empleadoid**, que oficiará de clave primaria.

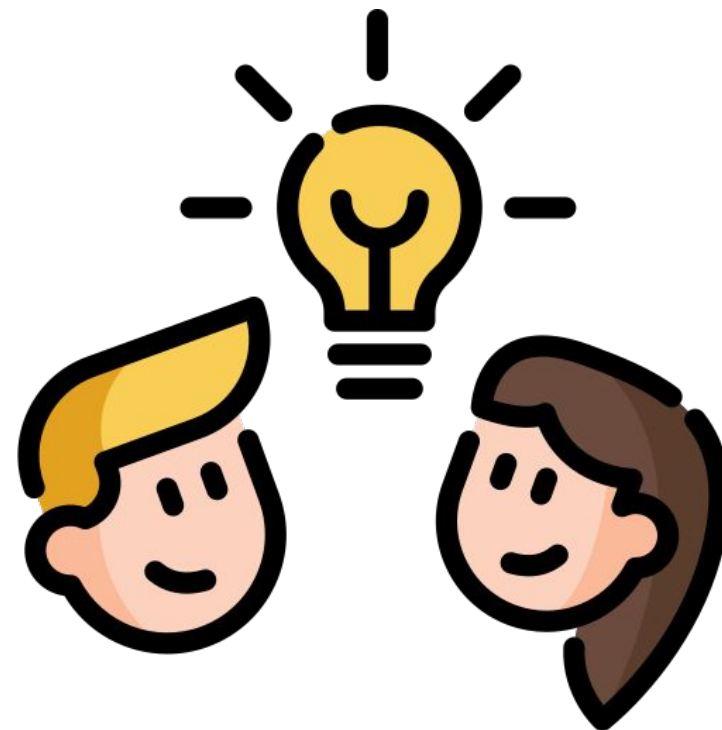
Pero, a su vez, contamos con otros campos, como ser, **legajo** y **documento** de identidad. Estos últimos, serán unívocos también, y pueden ser tranquilamente denominados como campos de clave candidata.



## Tipos de índices

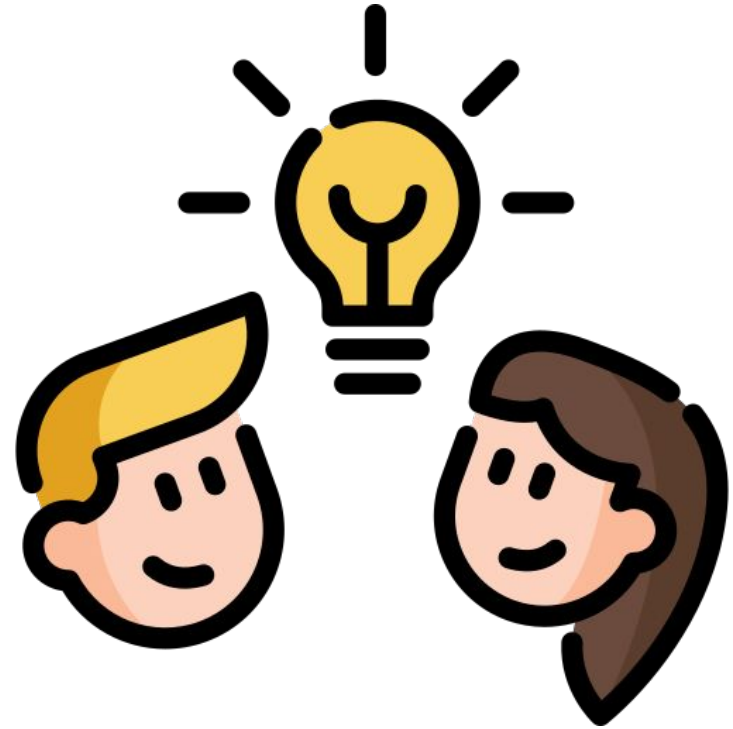
**En resumen, los índices son una parte esencial del rendimiento de una base de datos relacional, ya que ayudan a mejorar la velocidad y eficiencia de las consultas.**

**Además de esto, los índices son el punto clave que le dieron origen al desarrollo efectivo del modelo relacional. Este desarrollo cuenta con una serie de Formas normales, las cuales permiten definir un fundamento teórico de porqué debemos separar la información en dos o más tablas y por qué, cuanto más información tenemos, más tablas deben ser las que se originen a partir de la misma.**



## Tipos de índices

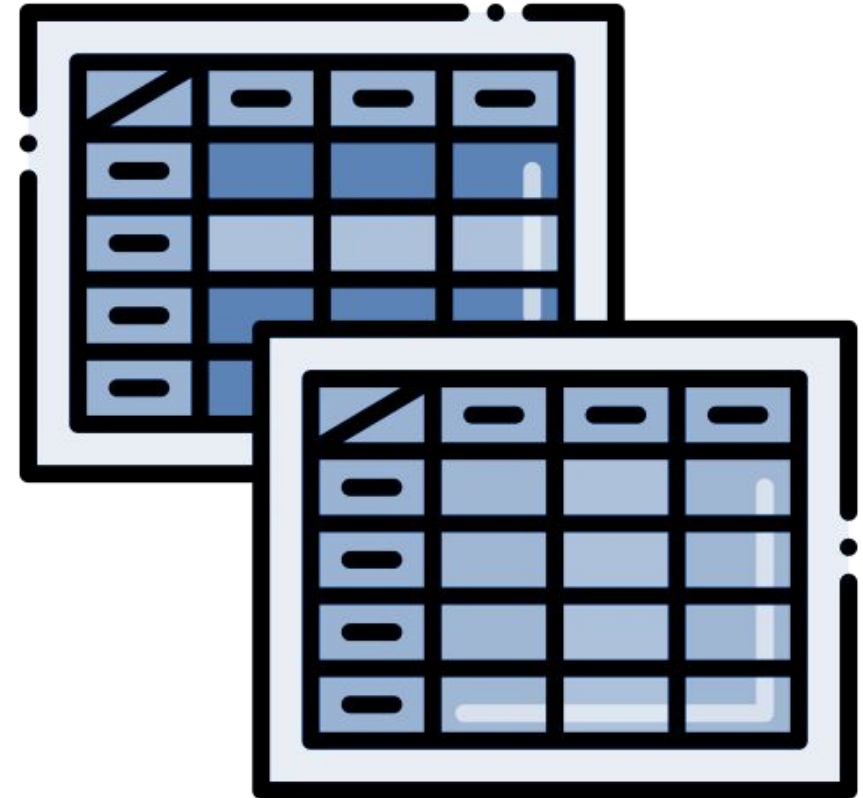
Si bien, cualquier persona tiende a pensar que, mientras más tablas tengamos en una bb.dd. esta será más grande y voluminosa, con el correr del tiempo y el incremento de la información histórica de cualquier sistema de software, el modelo relacional y el tipo de forma normal elegido hará que dicha base de datos sea más efectiva, rápida, y que su peso sea notablemente menor.



# Formas normales

# Formas Normales

En el contexto de bases de datos relacionales y SQL, las formas normales son una serie de reglas que se aplican para evitar redundancias y anomalías en los datos almacenados en las tablas de la base de datos.

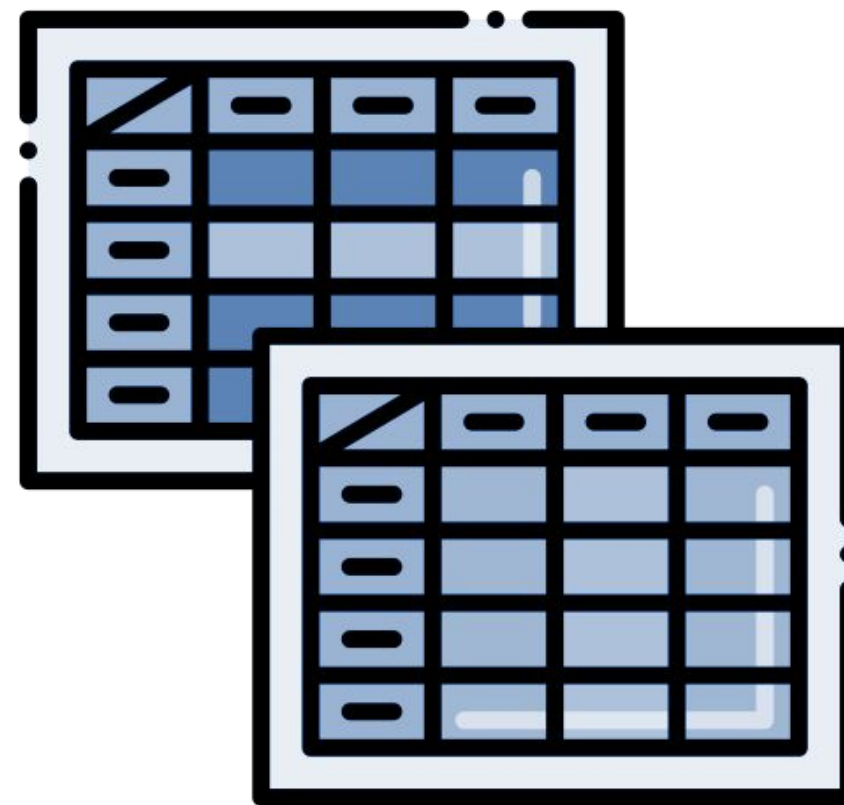


# Formas Normales






**Existen seis formas normales originales:**

- 1ª forma normal
- 2º forma normal
- 3º forma normal
- Forma normal de *Boyce-Codd*
- 4º forma normal
- 5º forma normal

Pero en la práctica, las tres primeras formas normales son las más utilizadas.



# Formas Normales

Result Grid					
Filter Rows: <input type="text" value="Search"/>					
Edit:   					
Export/Import:  					
id	nombre_alumno	apellido_alumno	nombre_materia	nombre_profesor	apellido_profesor
1	Juan	Pérez	Matemáticas	Carlos	González
2	María	García	Matemáticas	Carlos	González
3	Pedro	Fernández	Matemáticas	Carlos	González
4	Laura	López	Matemáticas	Carlos	González
5	Carlos	Ramírez	Historia	Ana	Sánchez
6	Lucía	Martínez	Historia	Ana	Sánchez
7	Miguel	Hernández	Historia	Ana	Sánchez
8	Ana	Díaz	Física	Javier	Muñoz
9	Sofía	Alvarez	Física	Javier	Muñoz
10	Diego	Gómez	Física	Javier	Muñoz

Esta tabla estructura la información de alumnos, materias y profesores. Como podemos apreciar, cumple con el criterio de una tabla plana, tal como vimos en la clase inicial dedicada a entender las tablas SQL.

Si debemos aplicarle a esta tabla, la primera forma normal, deberíamos dividir estos datos en tablas más pequeñas, de modo que no existan campos repetidos.

# Formas Normales

**ALUMNOS**

id	nombre	apellido
1	Juan	Pérez
2	María	García
3	Pedro	Fernández
4	Laura	López
5	Carlos	Ramírez
6	Lucía	Martínez
7	Miguel	Hernández
8	Ana	Díaz
9	Sofía	Alvarez
10	Diego	Gómez

**MATERIAS**

id	nombre
1	Matemáticas
2	Historia
3	Física

**PROFESORES**

id	nombre	apellido
1	Carlos	González
2	Ana	Sánchez
3	Javier	Muñoz

Basándonos en la **1ª Forma Normal**, deberíamos crear tres tablas separadas: una para **Alumnos**, otra para **Materias** y otra para **Profesores**. Luego, crearemos una cuarta tabla la cual se relaciona a las tres anteriores, almacenando la información correspondiente a las relaciones.



## Formas Normales

### AlumnosMateriasProfesores

	id	id_alumno	id_materia	id_profesor
▶	1	1	1	1
◀	2	2	1	1
	3	3	1	1
◀	4	4	1	1
	5	5	2	2
◀	6	6	2	2
	7	7	2	2
◀	8	8	3	3
	9	9	3	3
◀	10	10	3	3

La cuarta tabla, llamada eventualmente **AlumnosMateriasProfesores**, será la más simple de todas, porque solo contendrá información de cada uno de los índices primarios de las tres tablas restantes.

Si bien aquí la información no es concisa para nuestra vista, esta forma es totalmente válida y hará que al crecer la bb.dd en contenido, éste comience a ser más eficiente con el correr del tiempo y del crecimiento en volumen de datos.

## Segunda forma normal

## Formas Normales (segunda forma normal)

Para aplicar la segunda forma normal a los datos que vimos anteriormente, es necesario que identifiquemos a las dependencias funcionales para luego separarlas en tablas independientes.

En este caso, la tabla **AlumnosMateriasProfesores** tiene una dependencia funcional parcial en el campo **nombre\_materia** hacia el campo **nombre\_profesor**, ya que los nombres de los profesores se repiten, (*o pueden repetirse en un futuro*), para diferentes materias.

**MATERIAS**

id	nombre
1	Matemáticas
2	Historia
3	Física

**PROFESORES**

id	nombre	apellido
1	Carlos	González
2	Ana	Sánchez
3	Javier	Muñoz

## Formas Normales (segunda forma normal)

Por lo tanto, podemos separar esta dependencia funcional en una nueva tabla

**ProfesoresMaterias**, donde almacenaremos la relación entre los profesores y las materias que imparten.

De esta forma, ya sea para este modelo de datos o para un modelo de datos futuro, donde un profesor puede dictar más de una materia, la estructura de tablas de esta base de datos estará correctamente preparada.

**MATERIAS**

	id	nombre
▶	1	Matemáticas
	2	Historia
	3	Física

**PROFESORES**

	id	nombre	apellido
▶	1	Carlos	González
	2	Ana	Sánchez
	3	Javier	Muñoz

## Formas Normales (segunda forma normal)

Por lo tanto, podemos separar esta dependencia funcional en una nueva tabla

**ProfesoresMaterias**, donde almacenaremos la relación entre los profesores y las materias que imparten.

De esta forma, ya sea para este modelo de datos o para un modelo de datos futuro, donde un profesor puede dictar más de una materia, la estructura de tablas de esta base de datos estará correctamente preparada.

**MATERIAS**

id	nombre
1	Matemáticas
2	Historia
3	Física

**PROFESORES**

id	nombre	apellido
1	Carlos	González
2	Ana	Sánchez
3	Javier	Muñoz

**ProfesoresMaterias**

id_profesor	id_materia
1	1
1	2
2	2
1	3
3	3

**Otras formas normales**

## Otras Formas normales

Y para entender el resto de las formas normales, pensemos sobre lo trabajado hasta aquí. Solo con datos de alumnos, materias y profesores, terminamos creando 5 tablas.

Ahora, si además de registrar estos datos, debiéramos incluir el registro y control de **Aulas**, **Horarios**, **Matrículas**, y otros datos importantes de estudiantes y profesores, terminaremos sumergiéndonos seguramente en algún otro modelo relacional de la tercera o cuarta forma, o seguramente dentro de la forma *Boyce-Codd*.

**MATERIAS**

id	nombre
1	Matemáticas
2	Historia
3	Física

**PROFESORES**

id	nombre	apellido
1	Carlos	González
2	Ana	Sánchez
3	Javier	Muñoz

**PROFESORESMATERIAS**

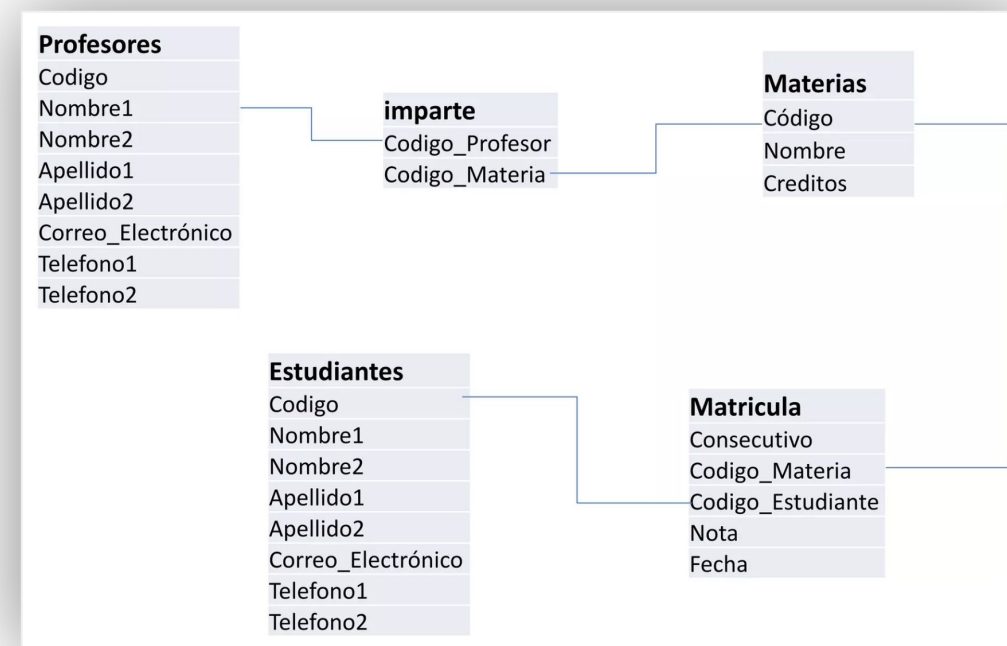
id_profesor	id_materia
1	1
1	2
2	2
1	3
3	3

**ALUMNOS**

id	nombre	apellido
1	Juan	Pérez
2	María	García
3	Pedro	Fernández
4	Laura	López
5	Carlos	Ramírez
6	Lucía	Martínez
7	Miguel	Hernández
8	Ana	Díaz
9	Sofía	Alvarez
10	Diego	Gómez

## Otras Formas normales

No es importante conocer en detalle todas estas formas normales, pero sí es clave que leamos sobre las mismas con tiempo, para así tenerlas presente y poder diseñar tablas en bases de datos, que sean eficientes y que podamos escalar en complejidad, mientras más datos simples necesitemos cruzar o almacenar en el modelo relacional que debemos construir.

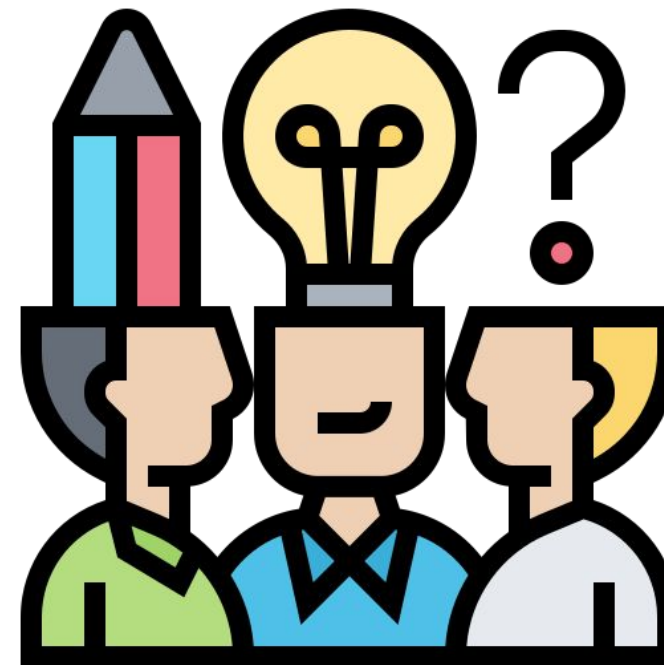




## Formas Normales

**El uso de las formas normales es algo cotidiano, que si bien no debemos aprenderlo de memoria, sí es bueno ir teniendo el concepto general en la cabeza.**

**Te recomendamos repasar lo explicado hasta aquí, de cara a que puedas afianzar más la importancia de las formas normales, ya que en nuestro próximo encuentro volcaremos la teoría a el uso práctico de formas normales en tablas relacionales.**

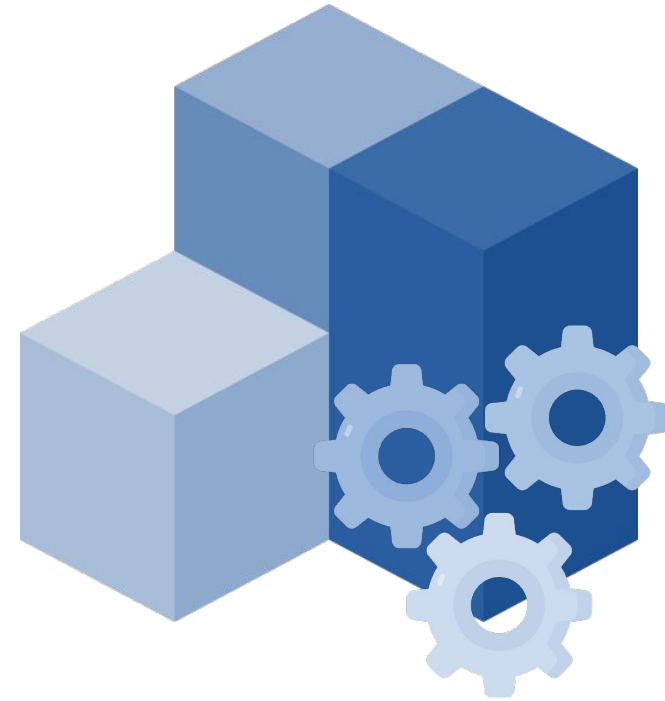


# Subconsultas SQL

## Subconsultas SQL

Las **Subconsultas** SQL son consultas insertas dentro de otra consulta. Son ideales para cuando debemos filtrar datos por algún valor específico, donde sí conocemos su valor coloquial, pero no conocemos o recordamos su ID.

Siempre es más común tener presente una descripción o nombre asociado, que un ID. Con esta información podemos filtrar los datos, aplicando una subconsulta SQL.



# Subconsultas SQL

Como muestra el ejemplo de código, una subconsulta es una nueva consulta SQL que puede ser ejecutada dentro de otra consulta SQL. Si te sientes como un personaje de la película **INCEPTION**, es normal... 🌀 ... igual, entender esto, es más fácil de lo que parece a simple vista.

```
Subconsultas SQL

SELECT id, campo1, campo2
FROM tabla
WHERE campo2 = (SELECT id2
                 FROM otraTabla
                 WHERE campob = 'valor');
```

# Subconsultas SQL

Como muestra el ejemplo de código, una subconsulta es una nueva consulta SQL que puede ser ejecutada dentro de otra consulta SQL.

Si te sientes como un personaje de la película **INCEPTION**, es normal ... 🌀 ... igual, entender esto, es más fácil de lo que parece a simple vista.

```
Subconsultas SQL

SELECT id, campo1, campo2
FROM tabla
WHERE campo2 = (SELECT id2
                 FROM otraTabla
                 WHERE campob = 'valor');
```

# Subconsultas SQL

Supongamos que tenemos una consulta como la siguiente, donde **campo2** equivale a una clave foránea la cual es una clave primaria (*numérica*) en **otraTabla**.

```
Subconsultas SQL

SELECT id, campo1, campo2
FROM tabla
WHERE campo2 = ...;
```

Para ver el valor de dicha clave numérica en **otraTabla**, ejecutamos una consulta aislada, estricta (uso de = ), la cual nos retornará un único registro y campo “**id2**”.

```
Subconsultas SQL

SELECT Id2 FROM otraTabla
WHERE campob = 'valor';
```

# Subconsultas SQL

Para resolver todo en un solo paso, incluimos la subconsulta SQL en búsqueda del ID correspondiente al valor coloquial que conocemos de memoria, encerrando la misma entre paréntesis.

Como esta subconsulta retorna un único registro y un único campo, ese valor será el que se aplique sobre el condicional establecido en la consulta principal.

```
Subconsultas SQL

SELECT id, campo1, campo2
FROM tabla
WHERE campo2 = (SELECT id2
                 FROM otraTabla
                 WHERE campob = 'valor');
```

# Subconsultas SQL

Finalmente, será como haber escrito el número en cuestión dentro del condicional de la primera consulta, pero sin tener que saber el valor numérico en cuestión.

El uso de paréntesis en subconsultas hace que estas se comporten igual que en el mundo de las matemáticas que alguna vez aprendimos en la escuela: primero se resuelven los resultados entre paréntesis, para que luego se resuelva el resto de la ecuación.

```
Subconsultas SQL

SELECT id, campo1, campo2
FROM tabla
WHERE campo2 = 3;
```



# Subconsultas SQL

**Veamos un ejemplo no tan abstracto:**

Aquí queremos obtener algunos campos de la tabla **Northwind.Products**, filtrando por un proveedor específico (**Suppliers**).

No recuerdo el código de **SupplierID**, pero sí recuerdo que la empresa se llama '*Exotic Liquids*'. De esta forma, armamos una subconsulta que nos retorne el **SupplierID** a partir del nombre del proveedor.

```
Subconsultas SQL

SELECT ProductID,
       ProductName,
       UnitPrice,
       UnitsInStock
FROM Northwind.Products
WHERE SupplierID = (SELECT SupplierID
                   FROM Suppliers
                   WHERE CompanyName = 'Exotic Liquids');
```

# Subconsultas SQL

También es totalmente válido utilizar en una subconsulta el condicional LIKE, aunque con este corremos riesgo de que nos retorne más de una coincidencia, lo cual puede no volver efectiva la consulta resultante.

```
Subconsultas SQL

SELECT ProductID,
       ProductName,
       UnitPrice,
       UnitsInStock
FROM Northwind.Products
WHERE SupplierID = (SELECT SupplierID
                   FROM Suppliers
                   WHERE CompanyName LIKE '%exotic%');
```

# Case When

## Case When

La cláusula **CASE** nos permite “*surfear*” entre diferentes condiciones y retornar un valor determinado cuando una de estas condiciones se cumple.

En el momento en el cual ocurre esto, la cláusula deja de comparar y devuelve el resultado indicado.



## Case When

Para que la cláusula **CASE** funcione de manera óptima, debemos combinarla con la sentencia **WHEN**.

Si ninguna de las condiciones se cumple, tenemos la posibilidad de definir un **ELSE** para retornar un valor por defecto.



## Case When

Aquí tenemos un ejemplo práctico,  
combinando CASE - WHEN - ELSE.

Queremos visualizar información puntual  
de la tabla Customers ordenando la  
misma por País (Country).

Si este campo es nulo o está vacío,  
entonces ordenamos por Ciudad.

```
Fetch  
  
SELECT CompanyName, City, Country  
FROM Customers  
ORDER BY  
    (CASE  
        WHEN Country IS NULL OR Country = "" THEN City  
        ELSE Country  
    END);
```

## Case When

El uso de **Case - When** puede aplicarse en diferentes lugares de una consulta de selección.

Por ejemplo, sobre tablas no normalizadas, podríamos reemplazar los campos con valores vacíos o nulos por información predeterminada.

En el siguiente ejemplo vemos cómo reemplazamos los datos vacíos en el campo **Region**, por un valor predeterminado: 'N/A'.

```
Fetch

SELECT EmployeeID,
        LastName,
        FirstName,
        Title,
CASE
    WHEN Region = '' THEN 'N/A'
    ELSE Region
END
    As Region
FROM Northwind.Employees;
```

## Case When

El uso de **Case - When** puede aplicarse en diferentes lugares de una consulta de selección.

Por ejemplo, sobre tablas no normalizadas, podríamos reemplazar los campos con valores vacíos o nulos por información predeterminada.

En el siguiente ejemplo vemos cómo reemplazamos los datos vacíos en el campo **Region**, por un valor predeterminado: 'N/A'.

```
Fetch

SELECT EmployeeID,
        LastName,
        FirstName,
        Title,
CASE
    WHEN Region = '' THEN 'N/A'
    ELSE Region
END
    As Region
FROM Northwind.Employees;
```



# Sección práctica

Nos piden generar una serie de reportes sobre la tabla Products de la base de datos Northwind.

Estos reportes son gerenciales, por lo tanto debemos destacar claramente la información que la gerencia desea conocer.

Veamos los requisitos a continuación:



# Prácticas

Necesitamos simplificar la visualización de datos de la tabla **Products**, presentando en una consulta de selección, los siguientes campos:

- **ProductID, ProductName, UnitPrice, UnitsInStock, ReorderLevel**

Sobre esta consulta de selección base, realiza las siguientes consignas:

1. Ejecuta una consulta de selección de todos estos datos, ordenando los mismos por:
  - a. **CategoryID, ProductName**
2. En una nueva consulta de selección con la base inicial:
  - a. Muestra una leyenda en el campo **ReorderLevel**, que diga *'Reponer Stock'*, en aquellos productos donde el campo **UnitsInStock** esté por debajo de **ReorderLevel**
  - b. Ordena los productos por **ProductName**
3. Ejecuta una consulta de selección igual al Punto 1, agregando la siguiente condición
  - a. **CategoryID = (el id de la categoría llamada 'Seafood')**
    - i. utiliza una subconsulta SQL en esta condición



# Muchas gracias.



Ministerio de Economía  
**Argentina**

Secretaría de  
Economía del Conocimiento

*primero  
la gente*