



Introducción a las bases de datos

Sofía Ortiz Valenzuela
Universidad Panamericana

Enero - Mayo 2022

SQL (Structured Query Language)

- La forma en la que es posible interactuar con los DBMS, es decir, ejecutar las operaciones que un DBMS ofrece es mediante un lenguaje de comunicación. En el caso SGBD Relacionales (basados en el Modelo Relacional de datos), el lenguaje utilizado es **SQL (Structured Query Language)**.
- Estos lenguajes deben así permitir las siguientes operaciones y se puede dividir en dos tipos: (a) Definición de estructura y (b) Manipulación de datos.
 - Lenguaje de Definición de Datos (LDD): Permite definir y manipular la estructura de bases de datos almacenadas en el SGBD. Es más utilizado por el diseñador de la DB.
 - Ejemplos de operaciones son:
 - ◆ Crear las bases de datos necesarias.
 - ◆ Eliminar bases de datos existentes
 - ◆ Reestructurar las bases de datos existentes.
 - ◆ Consultar acerca de la estructura de las bases de datos existentes
 - Lenguaje de Manipulación de Datos (LMD): Permite ingresar, modificar, eliminar y datos, así como realizar y controlar las transacciones.
 - Ejemplos de operaciones:
 - ◆ Insertar de datos en las bases de datos
 - ◆ Eliminar datos de las bases de datos
 - ◆ Modificar datos de las bases de datos
 - ◆ Consultar datos de las bases de datos.
- Es importante considerar que SQL no es el gestor de base de datos, es una de las herramientas que ofrece un manejador para permitir al usuario (persona, programa, api) ejecutar las operaciones que ofrece el gestor. Una analogía puede ser compararlo con una interfaz gráfica de usuario al dar clic en un botón se ejecuta una acción del sistema, al ejecutar una sentencia SQL se ejecuta una acción en el DBMS.

Algebra relacional

- Un **lenguaje de consultas** es un lenguaje en el que los usuarios solicitan información de la base de datos.
- Los lenguajes suelen clasificarse en:
 - **Lenguajes procedimentales**: El usuario indica al sistema que lleve a cabo una serie de operaciones en la base de datos para calcular el resultado deseado
 - **Lenguajes no procedimentales**: El usuario describe la información deseada sin dar un procedimiento concreto para obtener esa información.
- El lenguaje de **álgebra relacional** es un lenguaje procedimental.

Consiste en un conjunto de operaciones que toman una o dos relaciones como entrada y generan otra relación nueva como resultado. Está nueva relación a su vez puede usarse como entrada para una nueva operación.

- Los manejadores de bases de datos ofrecen el lenguaje SQL que incluye elementos del enfoque procedimental.
 - Dentro de un SMDB, las tablas resultantes de las operaciones de álgebra relacional sólo se generan en memoria temporal para visualizar los datos y operar con ellas, pero no se almacenan en memoria permanente a no ser se indique explícitamente.

Entender álgebra relacional es la clave para entender los lenguajes de consulta SQL.

Operaciones de selección

- La operación de selección, selecciona tuplas que satisfacen un predicado dado. Es decir, se usa una condición para seleccionar filas de una tabla.
 - Se usa la letra griega sigma minúscula (σ) para denotar la selección.
 - El predicado aparece como subíndice de σ y expresa una condición sobre los atributos de la relación.
 - La relación sobre la cual se aplica la operación, y que funciona como argumento, se indica entre paréntesis a continuación de σ .
- Consulta: Seleccionar los estudiantes del campus de Guadalajara (GDL)

Estudiante		
ID	Nombre	Campus
1	Diego Joel	MX
2	Juan Pérez	GDL
3	Jorge Gutiérrez	MX
4	Katie Lucía	MX

$\sigma_{\text{Campus} = \text{'GDL'}} (\text{Estudiante})$



select *
from estudiante
where campus = 'GDL'

Estudiante		
ID	Nombre	Campus
2	Juan Pérez	GDL

$\sigma_{\text{cond}} (\text{Tabla})$

select *
from *Tabla*
where *cond*

Operaciones de selección

- Se permiten las comparaciones que usan: =, <>, <=, >=, <, > en el predicado de selección. Además se pueden combinar varios predicados usando las conectivas: y (\wedge), o (\vee) y no (\neg)

$\sigma_{ID \geq 2}$ (Estudiante)

Estudiante		
ID	Nombre	Campus
1	Diego Joel	MX
2	Juan Pérez	GDL
3	Jorge Gutiérrez	MX
4	Katie Lucía	MX

$\sigma_{\text{Campus} = \text{'GDL'} \wedge ID \geq 2}$ (Estudiante)

Estudiante		
ID	Nombre	Campus
1	Diego Joel	GDL
2	Juan Pérez	GDL
3	Jorge Gutiérrez	MX
4	Katie Lucía	MX

σ_{cond} (Tabla)

Operación de proyección

- La operación de proyección es una operación unaria que devuelve la relación de argumentos, excluyendo algunos argumentos.
- Se denota por la letra griega mayúscula pi (Π)
- Se crea una lista de los atributos que desean que aparezca en el resultado como subíndice de Π .
- Consulta: Listar el nombre de los estudiantes

Estudiante		
ID	Nombre	Campus
1	Diego Joel	MX
2	Juan Pérez	GDL
3	Jorge Gutiérrez	MX
4	Katie Lucía	MX

$\Pi_{\text{Nombre}}(\text{Estudiante})$



select Nombre
from Estudiante

Estudiante
Nombre
Diego Joel
Juan Pérez
Jorge Gutiérrez
Katie Lucía

$\Pi_{a,b,\dots,n}(\text{Tabla})$

select a,b,...,n
from Tabla

Operación de unión

- Permite combinar los datos de dos relaciones.
- El resultado de la unión es una relación formada con las columnas de una de ellas y las filas de ambas relaciones, excluyendo las tuplas duplicadas.
- Las relaciones deben de ser compatibles en dominio y grado.

Table_A	
ID	Campus
1	MX
2	GDL
3	MX

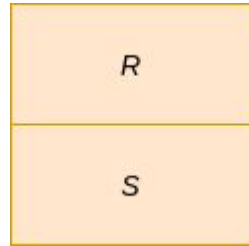
Table_B	
ID	Campus
1	MX
4	GDL

$$\Pi_{ID, Campus}(Table_A) \cup \Pi_{ID, Campus}(Table_B)$$

select * from Table_A
union
select * from Table_B

Table_C	
ID	Campus
1	MX
2	GDL
3	MX
4	GDL

R ∪ S

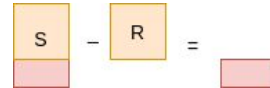


select * from R
union
select * from S

Operación de diferencia de conjuntos

- Denotada por $-$, permite hallar tuplas que están en una relación pero no en otra.
- La expresión $r - s$ da como resultado una relación que contiene las tuplas que están en r pero no en s .
- Las relaciones r y s deben de ser compatibles en grado y dominio.

$$R - S$$



$$\Pi_{ID, Nombre}(BaseDatos) - \Pi_{ID, Nombre}(CalculoDiferencial)$$

BaseDatos	
ID	Nombre
1	Ana Teresa
2	Andrés Gabriel
3	Rafael

$-$

CalculoDiferencial	
ID	Nombre
1	Ana Teresa
2	Andrés Gabriel



Resultado	
ID	Nombre
3	Rafael

`select * from BaseDatos
except
select * from CalculoDiferencial`

`select * from R
except
select * from S`

Operación de producto cartesiano

- La operación producto cartesiano se denota por una cruz (x), permite combinar la información de cualesquiera dos relaciones.
- El producto cartesiano de las relaciones r_1 y r_2 se escribe $r_1 \times r_2$.

Estudiante x Campus

Estudiante	
est_id	est_nombre
1	Ana Teresa
2	Andrés Gabriel
3	Rafael

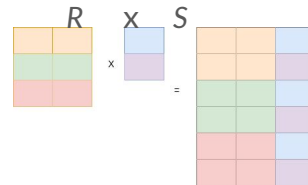
X

Campus	
campus_id	campus_nombre
1	CDMX
2	GDL



select *
from Estudiante, Campus

Resultado			
est_id	est_nombre	campus_id	campus_nombre
1	Ana Teresa	1	CDMX
1	Ana Teresa	2	GDL
2	Andrés Gabriel	1	CDMX
2	Andrés Gabriel	2	GDL
3	Rafael	1	CDMX
3	Rafael	2	GDL



select *
from R,S

Composición de operaciones relacionales

- El resultado de una operación de álgebra relacional es también una relación
- Considere consultar:
 - Obtener los nombres de estudiantes de campus guadalajara de la relación Estudiante

Estudiante		
ID	Nombre	Campus
1	Diego Joel	MX
2	Juan Pérez	GDL
3	Jorge Gutiérrez	MX
4	Katie Lucía	MX




Consulta:

$\Pi_{\text{Nombre}}(\sigma_{\text{Campus} = \text{'GDL'}}(\text{Estudiante}))$

Operaciones complementarias



- Las operaciones fundamentales de álgebra relacional son suficientes para expresar cualquier consulta de álgebra relacional.
 - Pero existen otras operaciones complementarias que permiten expresar de forma más sencillas algunas consultas que resultarían complicadas de expresar utilizando únicamente las operaciones fundamentales.
 - Para cada una de las operaciones complementarias es posible expresarlas mediante las operaciones fundamentales.
- 

Intersección de conjuntos

- Permite identificar tuplas que son comunes a dos relaciones.
- Al igual que la unión, las dos tablas deben tener el mismo grado y los dominios de sus atributos deben ser compatibles.
- El resultado de la intersección es el conjunto de tuplas que están tanto en R como en S.

BaseDatos	
ID	Nombre
1	Ana Teresa
2	Andrés Gabriel
3	Rafael

\cap

CalculoDiferencial	
ID	Nombre
1	Ana Teresa
2	Andrés Gabriel

```
select ID, Nombre
from BaseDatos
intersect
select ID, Nombre
from CalculoDiferencial
```

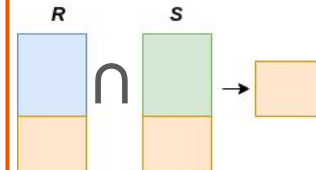


Resultado	
ID	Nombre
1	Ana Teresa
2	Andrés Gabriel

Otra forma de obtener la intersección es:

$$R \cap S = R - (R - S)$$

$R \cap S$



```
select * from R
intersect
select * from S
```

Reunión (join) natural

- Es una operación que permite combinar ciertas selecciones y un producto cartesiano en una sola selección. Se usa para conectar datos a través de distintas relaciones.
- Se denota por el símbolo reunión \bowtie .
- La operación reunión natural :
 - Forma un producto cartesiano de sus dos argumentos
 - Realiza una selección forzando la igualdad de los atributos que aparecen en ambos esquemas de la relación
 - Elimina los atributos duplicados.

$$r \bowtie s = \Pi_{R \cup S} (\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \dots \wedge r.A_n = s.A_n} (r \times s))$$

- Consulta: Obtener listado de estudiantes y las materias que cursa.

Estudiante		
estudiante_id	nom_estudiante	materia_id
1	Ana Teresa	1
2	Andrés Gabriel	1
3	Rafael	2

Materia	
materia_id	nom_materia
1	Base de datos
2	Cálculo diferencial
3	Probabilidad y Estadística



estudiante \bowtie materia

Resultado			
jugador_id	nom_estudiante	materia_id	nom_materia
1	Ana Teresa	1	Base de datos
2	Andrés Gabriel	1	Base de datos
3	Rafael	2	Cálculo diferencial

select *
from
estudiante
natural join
materia

La operación **JOIN** es de las más utilizadas, pero en la práctica se usan otros tipos de join.

select *
from tabla 1
natural join tabla 2

Θ - Join

- Es un producto cartesiano seguida de una selección:

a. $R \bowtie_{\Theta} S = \sigma_{\Theta} (R \times S)$

- Consulta: Obtener listado de estudiantes y las materias que cursa.

estudiante $\bowtie_{\text{materia_id} = \text{materia_id}}$ materia

Estudiante		
estudiante_id	nom_estudiante	materia_id
1	Ana Teresa	1
2	Andrés Gabriel	1
3	Rafael	2

Materia	
materia_id	nom_materia
1	Base de datos
2	Cálculo diferencial
3	Probabilidad y Estadística



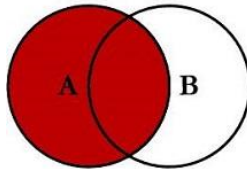
Resultado				
e.jugador_id	e.nom_estudiante	e.id	m,materia_id	m.nom_materia
1	Ana Teresa	1	1	Base de datos
2	Andrés Gabriel	1	1	Base de datos
3	Rafael	1	2	Cálculo diferencial

```
select *  
from estudiante e  
join materia m  
on e.materia_id = m.materia_id
```

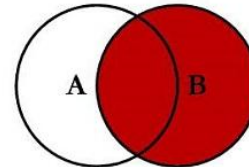
```
select *  
from tablaA a  
join tablaB b  
on a.campo =  
b.campo
```

Tipos de Joins

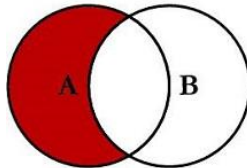
SQL JOINS



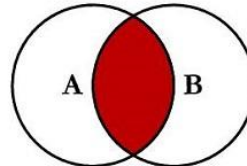
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



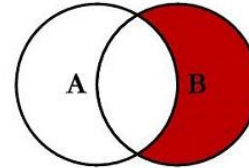
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



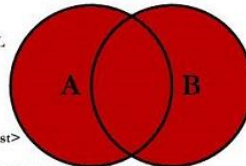
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



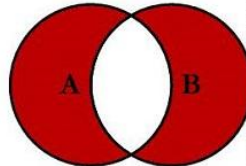
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

División

- La operación de **división**, denotada por \div , resulta adecuada para las consultas que incluyan la expresión “para todos”.
- Encuentra todos los valores del primer operando (tabla) que están relacionados con TODOS los valores del segundo operando (tabla).
- Sean $r(R)$ y $r(S)$ y $S \subseteq R$; es decir, todos los atributos del esquema S están también en el esquema R . La relación $r \div s$ es una relación del esquema $R - S$ (del esquema que contiene todos los atributos del esquema R que no están en S). Una tupla está en $r \div s$ si y sólo si se cumplen estas dos condiciones:

1. t está en $\Pi_{R-S}(r)$
2. Para cada tupla t_s de s hay una tupla t_r de r que cumple las dos condiciones siguientes:
 - a. $t_r[S] = t_s[S]$
 - b. $t_r[R - S] = t$

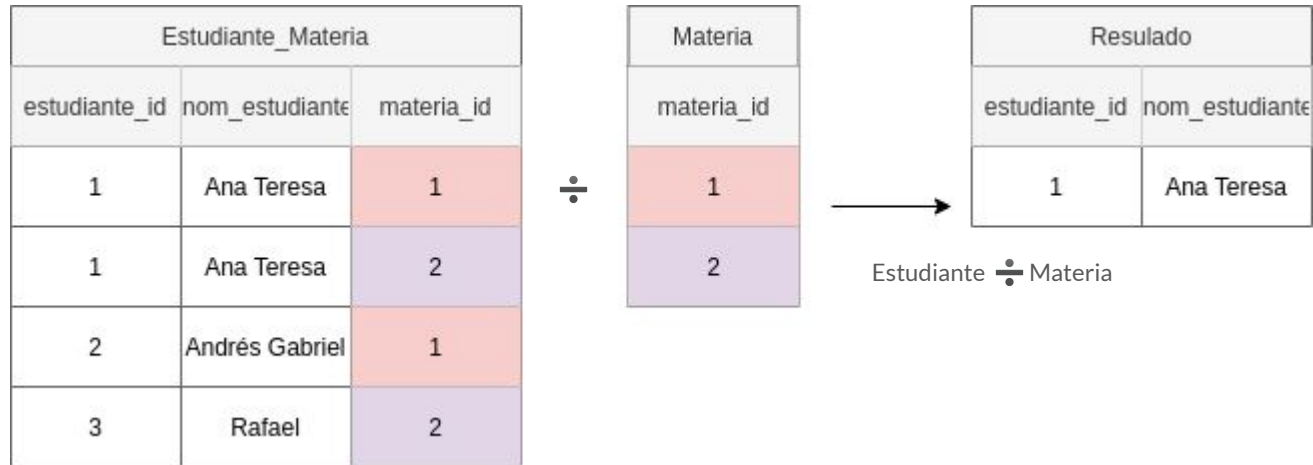
- Es posible definir la operación división en término de operaciones fundamentales.

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

División

Encuentra todos los valores del primer operando (tabla) que están relacionados con TODOS los valores del segundo operando (tabla).

- Consulta: Obtener los estudiantes que estén cursando las materias existentes en la tabla Materias.



No hay sentencia SQL esta operación se puede obtener el resultado mediante operaciones fundamentales

Concepto de normalización

- Cuando se diseña una base de datos mediante el modelo relacional, al igual que ocurre en otros modelos de datos, tenemos distintas opciones, es decir, podemos obtener diferentes esquemas relacionales, y algunos de ellos van a representar la realidad mejor que otros
- Con la teoría de normalización, se consigue una formalización en el diseño lógico de base de datos relacionales, lo que permite disponer de métodos de ayuda de diseño.



Normalizar las tablas de una base de datos **significa** optimizar su diseño para no repetir datos innecesariamente para prevenir problemas de inconsistencia y permita recuperar fácilmente los datos.


Concepto de normalización



- El proceso de diseño que hemos seguido es: Entidad Relación -> Esquema relacional, teniendo como resultado un conjunto de tablas, las cuales dependiendo del diseñador podrían presentar los siguientes problemas:
 - Incapacidad para almacenar ciertos hechos
 - Redundancia y, por tanto, posibilidad de inconsistencias
 - Ambigüedades
 - Pérdida de información
 - Pérdida de dependencias funcionales, es decir, de ciertas restricciones de integridad que dan lugar a interdependencias entre los datos.
 - Aparición, en la base de datos que no son válidos en el mundo real (anomalías de inserción, borrado y modificación)

Concepto de normalización



- E. F. Codd, que fue el creador de las bases de datos relacionales, en 1970 definió las tres primeras formas normales, que son:
 - 1 FN (primera Forma Normal)
 - 2 FN (segunda Forma Normal)
 - 3 FN (tercera Forma Normal)
 - Se dice que una tabla cumple con determinada forma normal cuando satisface las restricciones por dicha forma normal.
 - Durante el proceso de normalización hay que tener siempre en mente la posibilidad de descomponer una tabla en otras más pequeñas.
- 

Dependencia funcional

- Antes de iniciar con las formas normales, definiremos el concepto de dependencia funcional ya que la normalización se base en este concepto.
- Primero se definirá un esquema de una tabla $T(A,D)$, cómo un conjunto compuesto por:
 - Un conjunto de atributos A sobre los que se definen todas las *ocurrencias* del esquema, es decir, las tuplas.
 - Un conjunto de restricciones o dependencias D que debe satisfacer cualquier ocurrencia.

Dependencia funcional: Sea $X \subseteq A, Y \subseteq A$ dos atributos. Existe *Dependencia funcional*, $X \rightarrow Y$, de Y con respecto a X cuando a todo valor de X le corresponde uno y sólo uno de Y .

Se dice que un atributo o conjunto de atributos B *dependen funcionalmente* del atributo o conjunto de atributos A ($A \rightarrow B$), **si y sólo si a cada valor de A le corresponde un único valor de B** . Por ejemplo, si se tiene el CURP de una persona, se puede saber su fecha de nacimiento.

CURP \rightarrow Fecha de Nacimiento, Género

Las dependencias funcionales permiten expresar las restricciones que no se pueden expresar con super claves.

Las dependencias funcionales generalmente se definen por parte de *negocio*.

Transitividad

- **Transitividad de una dependencia funcional:** Si Y depende funcionalmente de X y Z depende funcionalmente de Y, entonces se cumple que Z depende funcionalmente de X:
$$(X \rightarrow Y) \text{ y } (Y \rightarrow Z) \Rightarrow (X \rightarrow Z)$$

Ejemplo: $(\text{Matrícula} \rightarrow \text{Grupo}) \text{ y } (\text{Grupo} \rightarrow \text{Aula}) \Rightarrow (\text{Matrícula} \rightarrow \text{Aula})$

Es decir, un atributo Z es transitivamente dependiente de otro X si se conoce por diferentes vías, una directamente y otra a partir de otro atributo intermedio Y.

Primera forma normal (1FN)

- Una tabla se encuentra en primera forma normal si y sólo si los valores que componen el atributo de una tupla son *atómicos*.
 - Los atributos atómicos son valores únicos y los más pequeños posible, es decir, indivisibles.
- Además tiene una clave primaria, atributos no nulos y no tiene tuplas repetidas.

Materiales		
COD_MAT	DESC.	MEDIDAS
039	TORNILLO	3,5,5,7
067	ARANDELA	2,5
461	BROCA	5,6,7,8,9



Mat-Medidas		
COD_MAT	DESC.	DESC
039	TORNILLO	3.5
039	TORNILLO	5
039	TORNILLO	7
067	ARANDELA	2
067	ARANDELA	5
...		...



Mat-Medidas		
ID	COD_MAT	DESC
1	039	3.5
2	039	5
3	039	7
4	067	2
5	067	5
...

Materiales	
COD_MAT	DESC
039	TORNILLO
067	ARANDELA
461	BROCA

Grupos repetidos

Segunda forma normal (2FN)

- Una tabla se encuentra en segunda forma normal cuando está en la primera forma normal (1FN)
- Y, cada campo secundario (aquel que no pertenece a la clave principal) depende de la clave principal en su totalidad y no de una parte de ella.
 - Si la clave principal es simple entonces la tabla ya se encuentra en 2FN (si se encuentra en 1FN)

Para esta forma normal se usan las dependencias funcionales

Prestamo			
<u>ID_USR</u>	<u>C_LIBRO</u>	TITULO	FECHA_PRESTAMO
1	3421	Fisica I	02/04/2021



Prestamos		
ID_USR	C_LIBRO	FECHA_PRESTAMO
1	3421	02/04/2021

Libros	
C_LIBRO	TITULO
3421	Fisica I

Tercera forma normal (3FN)

- Una tabla se encuentra en la tercera forma normal cuando está en la segunda forma normal (2FN)
- Y, cada campo que no sea llave primaria sólo depende de la llave primaria o de las claves secundarias de la tabla y no depende de otro campo de tal forma que no “existen atributos no primario que son *transitivamente dependientes* de cada posible clave de la tabla”.
- El objetivo de la tercera forma normal es eliminar cualquier dependencia transitiva. Una dependencia transitiva implica que se puede saber un campo secundario a través de otro que no es la clave principal.

Ordenes					
ID_Orden	Fecha	ID_Cliente	ApellidoP	Nombre	Telefono
15	2021-01-25	038	Peréz	José	55962381
17	2021-02-02	124	Molina	Eduardo	55181553

Ordenes		
ID_Orden	Fecha	ID_Cliente
15	2021-01-25	038
17	2021-02-02	124



Clientes			
ID_Cliente	ApellidoP	Nombre	Telefono
038	Peréz	José	55962381
124	Molina	Eduardo	55181553

Reglas de las formas normales

Primera forma normal (1FN):

La 1FN prohíbe los grupos repetidos, por lo tanto tenemos que convertir a la primera forma normal. Los pasos a seguir son:

- Tenemos que eliminar los grupos repetidos.
- Tenemos que crear una nueva tabla con la PK de la tabla base y el grupo repetido.

Segunda forma normal (2FN):

Se deben de eliminar cualquier columna no llave que no dependa de la llave primaria de la tabla. Los pasos a seguir son:

- Determinar cuáles columnas que no son llave no dependen de la llave primaria de la tabla.
- Eliminar esas columnas de la tabla base.
- Crear una segunda tabla con esas columnas y la(s) columna(s) de la PK de la cual dependen.

Tercera forma normal (3FN):

La tercera forma normal nos dice que tenemos que eliminar cualquier columna no llave que sea dependiente de otra columna no llave. Los pasos a seguir son:

- Determinar las columnas que son dependientes de otra columna no llave.
- Eliminar esas columnas de la tabla base.
- Crear una segunda tabla con esas columnas y con la columna no llave de la cual son dependientes.

Restricciones de integridad

- El término **integridad** en las bases de datos se refiere a **asegurar que los datos sean válidos**.
- “Las restricciones de integridad garantizan que las modificaciones realizadas en la base de datos por los usuarios autorizados no den lugar a una pérdida de la consistencia de los datos. Por tanto, **las restricciones de integridad integridad protegen contra daños** accidentes a las bases de datos” [Silberschatz, 2007].
- Algunos ejemplo de restricciones de integridad son:
 - El saldo de las cuentas no puede ser nulo.
 - No puede haber dos cuentas con el mismo número.
 - Todos los números de cuenta de la relación *impositor* deben tener el número de cuenta correspondiente en la relación *cuenta*.
 - El salario por hora de los empleados del banco debe ser, como mínimo, de 6 Euros la hora.
- **Las restricciones de integridad pueden ser predicados arbitrarios** que hagan referencia a la base de datos. Sin embargo, puede resultar costosa la comparación de estos predicados arbitrarios. Por tanto, la mayor parte de los sistemas de bases de datos permiten especificar restricciones de integridad que puedan probarse con una sobrecarga mínima.

También se conoce como restricciones de dominio, porque son restricciones a los valores que puede tomar un campo

Restricciones de integridad

- Se definen al momento de crear una tabla en la sentencia **CREATE TABLE**.
- Entre las restricciones de integridad se encuentran:
 - **not null:**
 - Prohíbe la inserción de valores nulos.
 - Aplica para los campos que son llave primaria que no deben tener valores nulos.
 - Campos que no son llave primaria que no deben tener un valor nulo, por ejemplo un estudiante con el campo 'nombre' en nulo.
 - Si se hace una inserción con un valor nulo en un campo con esta restricción el sistema manejará error.
 - **Sintaxis:** nombre varchar(50) not null
 - **unique** ($A_{j1}, A_{j2}, \dots, A_{jn}$)
 - Indica que los atributos $A_{j1}, A_{j2}, \dots, A_{jn}$ forman una clave candidata; es decir, ningún par de tuplas de la relación puede ser igual en todos los atributos.
 - Cuando se aplica sobre un sólo campo, indica que dicho campo no puede tener valores repetidos.
 - Permite que los valores de la clave candidata tenga valores nulos, a menos que se hayan declarado explícitamente cómo not null.
 - **Sintaxis:** email VARCHAR (50) UNIQUE

Restricciones de integridad

- **check:**

- Puede aplicarse sobre una columna o un dominio. La cláusula `check(P)` especifica un predicado P que deben cumplir todas las tuplas de la relación.
- Un uso frecuente de la cláusula `check` es garantizar que los valores de los atributos cumplan las condiciones especificadas. En postgres, mysql, etc; se usan las cláusulas `CONSTRAINT` y `CHECK`

```
CREATE TABLE tabla (  
    clave      int,  
    promedio float,  
    CONSTRAINT promedio_valido CHECK (promedio >= 0 and promedio <=10)  
);
```

La cláusula `CONSTRAINT promedio_valido` se emplea para dar el nombre 'promedio_valido' a la restricción.

Restricciones de integridad referencial

- Aplica para las llaves foráneas. El valor de una llave foránea en una tabla dependiente puede ser nulo o debe corresponder a uno de los valores de una llave primaria de la tabla maestra con cuál se establece la relación.
 - La restricción nos permite no insertar registros en la tabla dependiente con valores que no existen en la tabla primaria. Por ejemplo:
 - Estudiante (idEstudiante, idCampus) : No deberíamos poder insertar un registro con un campus que no esté registrado en la tabla Campus, ya que nos llevaría a inconsistencias
 - Campus (idCampos, nombre)
- Para expresar la integridad referencial se utilizan las cláusulas: REFERENCES y FOREIGN KEY

```
CREATE TABLE contacts(  
  contact_id INT NOT NULL,  
  customer_id INT,  
  contact_name VARCHAR(255) NOT NULL,  
  PRIMARY KEY(contact_id),  
  CONSTRAINT fk_customer  
    FOREIGN KEY(customer_id)  
      REFERENCES customers(customer_id)  
);
```

Restricciones de integridad referencial

- Cuando se viola una restricción de integridad referencial, el procedimiento es rechazar la acción que ha causado la violación.
- ¿En que escenarios se puede violar una restricción?
 - a) Se desea insertar registros en la tabla dependiente que no estén en la tabla primaria.
 - b) Se desea borrar registros en la tabla primaria que tengan asociaciones en la tabla secundaria.
 - c) Se desea actualizar la llave primaria en la tabla primaria que tengan asociaciones en la tabla secundaria.

Para los escenarios b) y c) se pueden manejar acciones:

- a) SET NULL => Permitir borrado y asignar nulo la llave foránea en la tabla dependiente
- b) RESTRICT => No permitir borrado
- c) CASCADE => Borrar información en cadena, en tabla principal y dependiente

Ejemplo de estructura de sentencia en postgres tomado de:
<https://www.postgresqltutorial.com/postgresql-foreign-key/>

```
[CONSTRAINT fk_name]
FOREIGN KEY(fk_columns)
REFERENCES parent_table(parent_key_columns)
[ON DELETE delete_action]
[ON UPDATE update_action]
```

Resumen restricciones



- Nos ayudan a minimizar los errores de captura de los datos y aseguran que los cambios realizados a una base de datos no provoquen inconsistencias en la información.
- ¿Que tenemos?
 - Restricciones de integridad o dominio: aplican sobre los valores de una tabla.
 - Restricciones de integridad referencial:
 - Aplican sobre las llaves foráneas y no permiten introducir valores que no existan en una tabla primaria a la cual se hace referencia.
 - No permiten el borrado de registros en la tabla primaria que tenga asociaciones en una tabla dependiente.

SQL (Structured Query Language)

- La forma en la que es posible interactuar con los DBMS, es decir, ejecutar las operaciones que un DBMS ofrece es mediante un lenguaje de comunicación. En el caso SGBD Relacionales (basados en el Modelo Relacional de datos), el lenguaje utilizado es **SQL (Structured Query Language)**.
- Estos lenguajes deben así permitir las siguientes operaciones y se puede dividir en dos tipos: (a) Definición de estructura y (b) Manipulación de datos.
 - Lenguaje de Definición de Datos (LDD): Permite definir y manipular la estructura de bases de datos almacenadas en el SGBD. Es más utilizado por el diseñador de la DB.
 - Ejemplos de operaciones son:
 - ◆ Crear las bases de datos necesarias.
 - ◆ Eliminar bases de datos existentes
 - ◆ Reestructurar las bases de datos existentes.
 - ◆ Consultar acerca de la estructura de las bases de datos existentes
 - Lenguaje de Manipulación de Datos (LMD): Permite ingresar, modificar, eliminar y datos, así como realizar y controlar las transacciones.
 - Ejemplos de operaciones:
 - ◆ Insertar de datos en las bases de datos
 - ◆ Eliminar datos de las bases de datos
 - ◆ Modificar datos de las bases de datos
 - ◆ Consultar datos de las bases de datos.
- Es importante considerar que SQL no es el gestor de base de datos, es una de las herramientas que ofrece un manejador para permitir al usuario (persona, programa, api) ejecutar las operaciones que ofrece el gestor. Una analogía puede ser compararlo con una interfaz gráfica de usuario al dar clic en un botón se ejecuta una acción del sistema, al ejecutar una sentencia SQL se ejecuta una acción en el DBMS.

SQL (Structured Query Language)

- Inserción:

<https://www.postgresqltutorial.com/postgresql-insert/>
<https://www.postgresql.org/docs/8.2/sql-insert.html>

```
INSERT INTO table_name(column1, column2, ...) VALUES (value1, value2, ...);
```

- Modificación :

```
UPDATE table_name  
SET column1 = value1,  
    column2 = value2,  
    ...  
WHERE condition;
```

<https://www.postgresql.org/docs/9.1/sql-update.html>
<https://www.postgresqltutorial.com/postgresql-update/>

- Borrado de registros

```
DELETE FROM table_name WHERE condition;
```

<https://www.postgresql.org/docs/10/sql-delete.html>
<https://www.postgresqltutorial.com/postgresql-delete/>

SQL (Structured Query Language)

- Consultas simples: Consultar tarea de álgebra relacional.
- Ordenamiento

```
select * from videojuego.partida order by fecha_inicio desc;
```

<https://www.postgresql.org/docs/9.5/queries-order.html>
<https://www.postgresqltutorial.com/postgresql-order-by/>

- Campos calculados

```
select count(*), max(fecha_inicio), min(fecha_inicio)  
from videojuego.partida;
```

- Agrupamientos: <https://www.postgresql.org/docs/9.4/tutorial-agg.html>

```
select nombre, count(*)  
from videojuego.partida p  
join videojuego.jugador_partida jp on p.partida_id = jp.partida_id  
join videojuego.jugador j on jp.jugador_id = j.jugador_id  
group by nombre  
order by nombre;
```

Referencias



- De Miguel A., Piattini M., “Fundamentos y Modelos de Bases de Datos”. 2ª ed., Alfaomega-Ra-ma, México, 2004.
- Silberschatz, A. Fundamentos de bases de datos (5a. ed.). Madrid etc: McGraw-Hill España, 2006. p.
<https://elibro.up.elogim.com/es/ereader/upanamericana/50087?page=203>