Integridad y modelos de datos

Tutoría

{desafío} latam_

Ideas fuerza

*La **integridad de datos** consiste en contar con datos **correctos** y que además estén completos.*

*Una **clave primaria** puede agregarse para un registro y asegurar que no pueda repetirse, garantizando la corrección de los datos.*

*Una **clave foránea** nos permite evitar el borrado de datos particulares, y garantizar así que permanezcan completos.*

*Un **modelo físico de datos** nos permite representar las relaciones entre los datos y tablas que presentan diferentes **cardinalidades**. Con ello, se puede implementar una **base de datos**.*

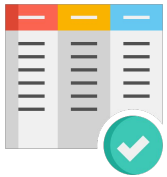
Recursos asincrónicos

- ¿Revisaste los recursos asincrónicos?
- ¿Alguno de ellos te dejó dudas?

Integridad de datos

Motivación

Integridad de datos = datos correctos + datos completos



- Problemas de diseño de una base de datos pueden causar problemas de integridad.
- En esta sesión aprenderemos a evitar algunos de ellos.

Integridad de datos

Asegurándose que el valor sea único y no nulo

Para asegurarnos que un valor realmente no pueda repetirse, ya que sería una amenaza a la integridad de los datos, especificaremos que esta columna es una **clave primaria (primary key, abreviado PK)**.

Id (PK)	Nombre	Edad
1	Consuelo	27
2	Consuelo	32
3	Francisco	27

Integridad de datos

Agregamos un identificador único

Para crear desde cero una tabla con PK	Para alterar una tabla existente

`CREATE TABLE users(`
`  id int primary key,`
`  nombre varchar,`
`  edad int`
`);`
	`/* Si ya tiene una columna para la PK */`
	`ALTER TABLE users(`
	`  ADD PRIMARY KEY (id)`
	`)`
	
	`/* Si no tiene la columna */`
	`ALTER TABLE users(`
	`  ADD COLUMN ID PRIMARY KEY`
	`)`

Esto no hay que saberlo de memoria, es fácil de encontrar en Google.

{desafío} latam_

Integridad de datos

Agregando una clave foránea

Una clave foránea (Foreign Key y abreviado FK) nos permite protegernos de esta situación. Una clave foránea debe apuntar a una clave única (y de preferencia primaria).

```
CREATE TABLE pagos (
  id int primary key,
  monto int,
  usuario_id integer references key users(id)
);
```

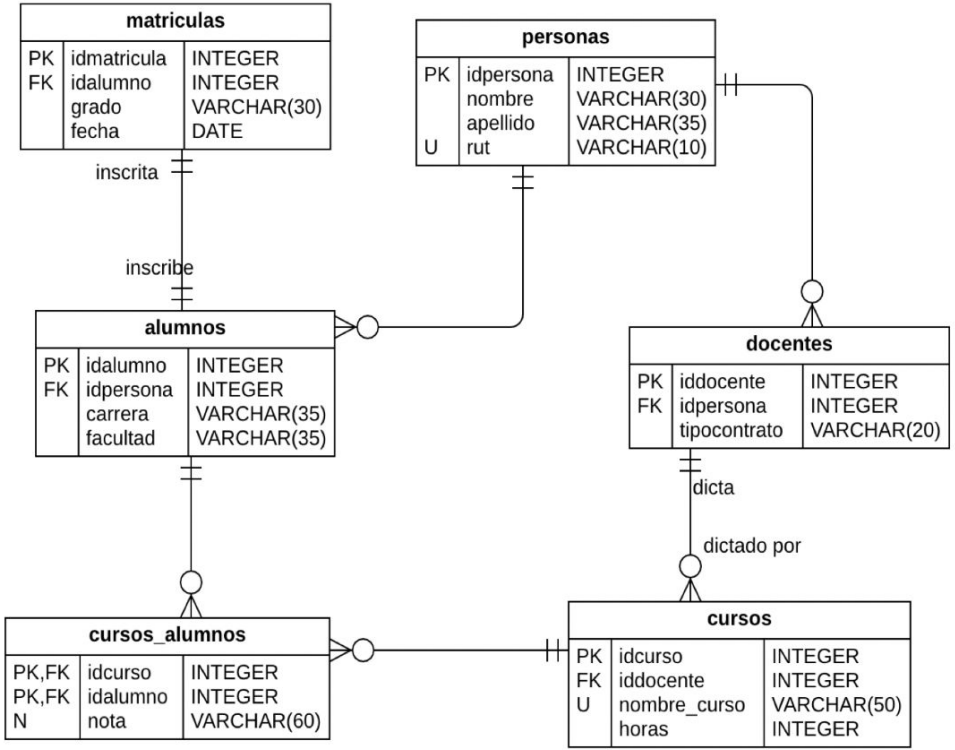
- Insertemos un pago asociado al usuario 1 y luego intentemos borrar el usuario 1.

/* Modelos de datos*/

Modelos de datos

Modelo físico

- El modelo físico de datos representa cómo se construirá el modelo de la base de datos.
- Aquí se especifican las tablas, columnas, claves primarias y foráneas, así como otras restricciones.

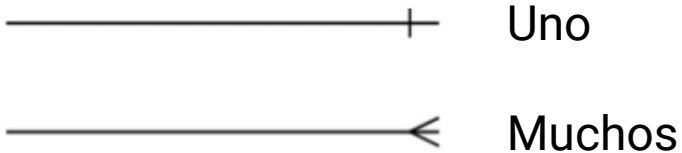


{desafío} latam_

Modelos de datos

Relaciones

Por ahora nos enfocaremos en las relaciones de uno y de muchos.

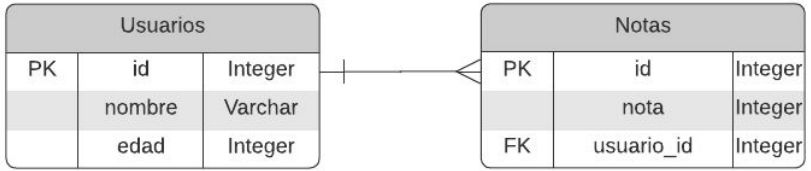


Modelos de datos

Cardinalidad

Se llama cardinalidad al tipo de relación que hay entre dos tablas. Principalmente, hablaremos de 3 casos:

1. 1 : 1 (Uno a uno)
2. 1 : N (Uno a muchos)
3. N : N (Muchos a muchos)



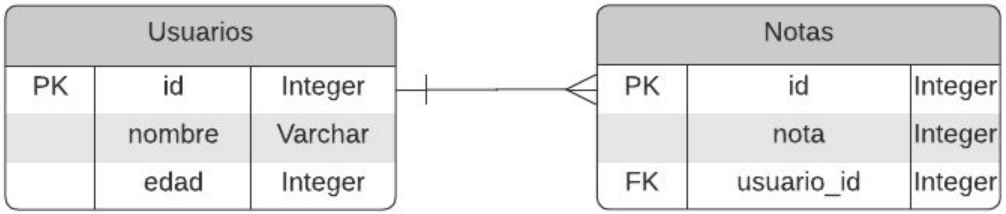
En nuestro ejemplo, tenemos una tabla de uno a muchos:

- Un usuario tiene muchas notas y una nota le pertenece a un único usuario.

Modelos de datos
Cardinalidad

Las tablas más utilizadas son las de 1 a N y las de N a N. Una tabla de 1 a N se implementa directamente tal como hemos hecho hasta ahora.

1. 1 : N (Uno a muchos)

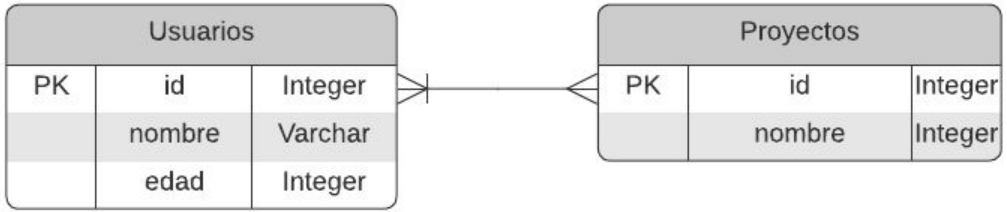


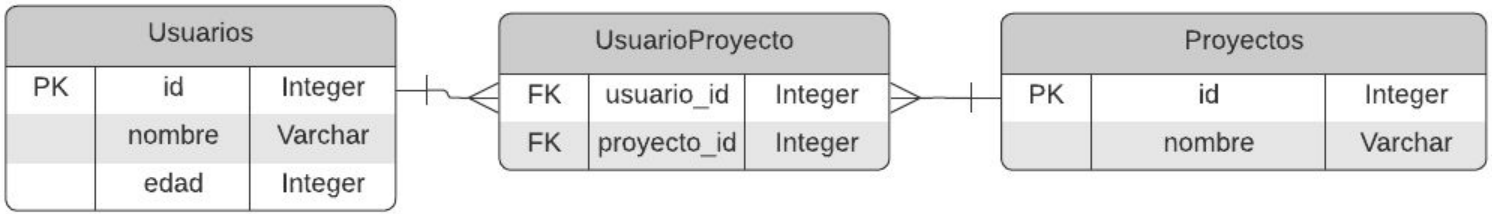
- La línea vertical del lado de la tabla usuario nos indica que necesariamente una nota le debe pertenecer a un usuario, ya que no puede haber una nota sin usuario.

Modelos de datos

Cardinalidad

Las relaciones N a N no pueden implementarse directamente, requieren de crear una tabla intermedia.

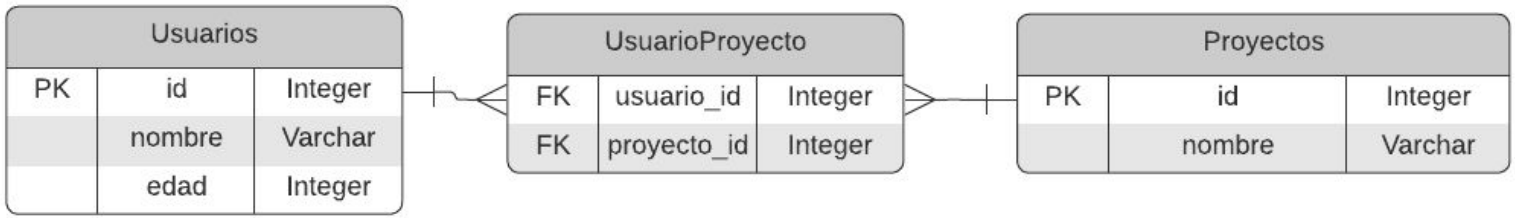




En algunos casos las veremos ya separadas en los modelos.

Modelos de datos

N a N y tabla intermedia

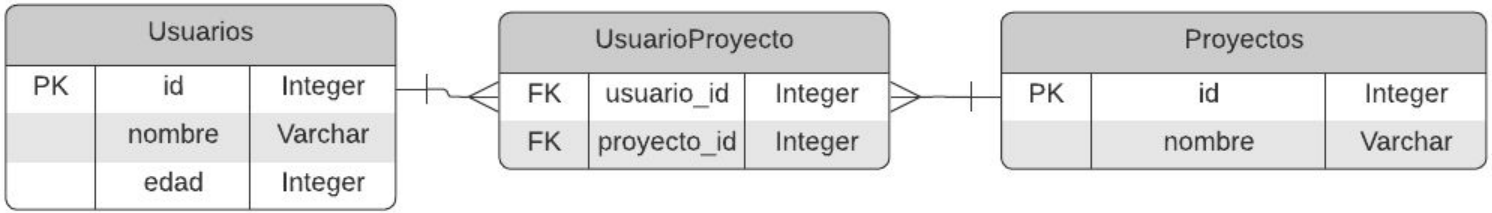


La lógica de la tabla intermedia siempre es igual entre 2 tablas de N a N: contiene las FK de las otras dos tablas y siempre queda del lado de los muchos.

Modelos de datos

N a N y tabla intermedia

Una vez que tenemos el modelo de esta forma podemos implementarlo con SQL.




```
CREATE TABLE
"Usuarios" (
"id" Integer,
"nombre" Varchar,
"edad" Integer,
 PRIMARY KEY ("id")
);
```

```
CREATE TABLE "UsuarioProyecto"
(
 "usuario_id" Integer,
 "proyecto_id" Integer,
 FOREIGN KEY ("usuario_id")
 REFERENCES ("Usuarios"("id"),
 FOREIGN KEY ("usuario_id")
 REFERENCES "Proyectos"("id")
);
```

```
CREATE TABLE "Proyectos" (
"id" Integer,
"nombre" Varchar,
PRIMARY KEY ("id")
);
```

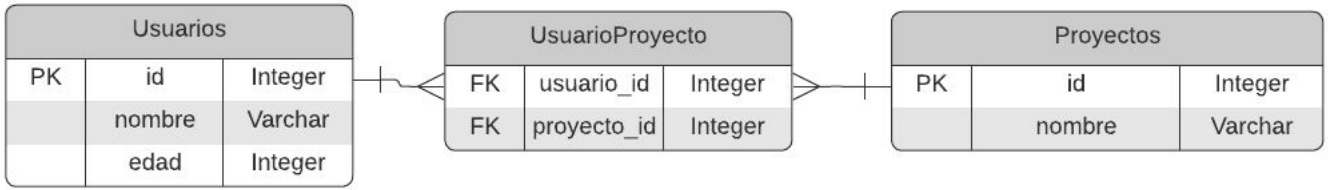
Ejercicio:
“Practicando la integridad referencial”

{desafío}
latam_

Practicando la integridad referencial

Contexto

A continuación, realizaremos un ejercicio donde pondremos en práctica y repasaremos los conceptos asociados a la integridad referencial. Para ello utilizaremos el siguiente modelo de datos:



Practicando la integridad referencial

Paso a paso

1. Inserta al menos 5 datos en las tablas usuarios y proyectos.
2. Intenta borrar proyecto de un usuario que no existe.
3. Crea una restricción que impida insertar proyectos sin el campo nombre.
4. Por cada usuario, en la tabla usuarios, cuenta cuántos de ellos tienen proyectos asignados.

Prueba - SQL

Prueba
SQL

- Descarga el archivo “Desafío”.
- Tiempo de desarrollo asincrónico: desde 4 horas.
- Tipo de desafío: individual.
- Revisemos los requerimientos desde plataforma.

¡AHORA TE TOCA A TI! 💪

{desafío} latam_

{desafío}
latam_

Academia de
talentos digitales