



Recibe una cálida:

# ¡Bienvenida!

---

Te estábamos esperando 😊 

# ➤ Sentencias para la manipulación de datos y transaccionalidad

---

**Plan formativo:** Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

# HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



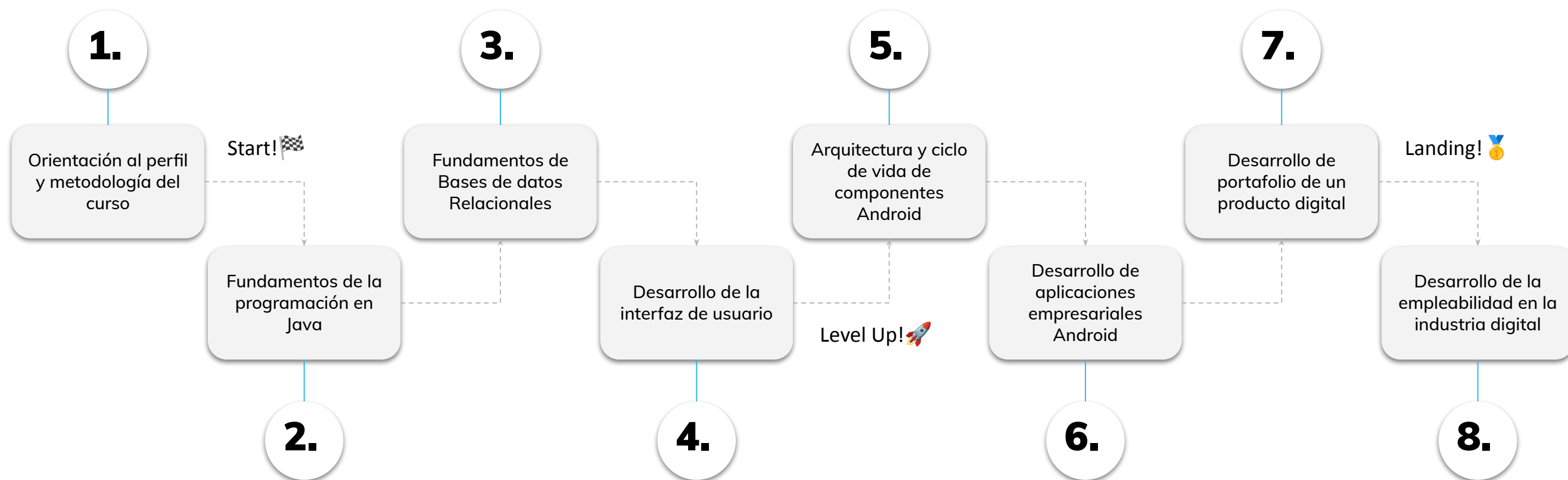
# ➤ Sentencias para la manipulación de datos y transaccionalidad

---

**Plan formativo:** Desarrollo de Aplicaciones Móviles Android Trainee V2.0

# HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



# REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Actualizar datos de una tabla utilizando DML
- ✓ Eliminar datos de una tabla utilizando DML

# LEARNING PATHWAY

3.

Start! 🏁

## Fundamentos de bases de datos relacionales

El objetivo de este tema es brindar una descripción básica de los conceptos asociados a las bases de datos relacionales que te permitirán desde tu ordenador instalar las herramientas necesarias para establecer una conexión a BD y comenzar a manipular dichos datos.

Integridad referencial

Insertar datos con integridad referencial

Actualizar datos con integridad referencial

Eliminar datos con integridad referencial

Insertión de datos con integridad referencial

Actualización de datos con integridad referencial

Eliminación de datos con integridad referencial

# OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



***Conocer el concepto de integridad referencial***



***Eliminar datos con integridad referencial***



***Actualizar datos con integridad referencial***





# › Integridad referencial



# Integridad referencial



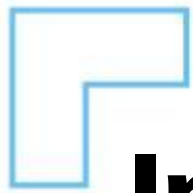
## ¿Qué es la integridad referencial?

La integridad referencial en SQL es un concepto que garantiza que las relaciones entre tablas en una base de datos se mantengan consistentes y precisas. **Se utiliza para asegurar que las relaciones entre tablas se mantengan válidas, evitando inconsistencias y referencias a registros inexistentes.**



En esencia, la integridad referencial asegura que los valores en una columna (clave foránea) de una tabla coincidan con los valores en la columna (clave primaria) de otra tabla





# Integridad referencial



## Usos de la integridad referencial:

1. Mantenimiento de la coherencia de datos: La integridad referencial garantiza que no se puedan crear relaciones incorrectas entre tablas, lo que asegura que los datos relacionados sean consistentes y precisos.
2. Prevención de acciones no deseadas: Evita la eliminación o actualización de registros en tablas relacionadas que podrían afectar la coherencia de los datos.
3. Mantener la estructura de la base de datos: La integridad referencial contribuye a la organización y estructura de la base de datos, al definir cómo las tablas se relacionan entre sí.



# Integridad referencial

## ✕ Implementación de la integridad referencial:

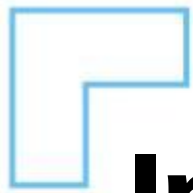
La integridad referencial se implementa a través de la definición de restricciones en las tablas que establecen cómo deben comportarse las relaciones.

## Las restricciones más comunes son:

Clave Primaria (PRIMARY KEY): Define la columna o conjunto de columnas que actúan como clave primaria en una tabla. Cada valor en la clave primaria debe ser único

Clave Foránea (FOREIGN KEY): Define una columna que establece una relación con la clave primaria de otra tabla.





# Integridad referencial

## Ejemplo de implementación:

Supongamos que tienes dos tablas:  
"Clientes" y "Pedidos".

La tabla "Pedidos" tiene una clave foránea  
"id\_cliente" que se relaciona con la clave  
primaria "id\_cliente" en la tabla "Clientes".

Si intentas insertar un pedido con un  
"id\_cliente" que no existe en la tabla  
"Clientes", la integridad referencial evitará  
esta acción.

Pedidos

id_orden	id_cliente
1	3

Clientes

id_cliente	nombre
3	Sofía



# › Insertar datos con integridad referencial



# Insertar datos con integridad referencial



## ¿Cómo insertar datos con integridad referencial?:

Insertar datos con integridad referencial en SQL implica asegurarse de que los valores que ingreses en una columna con clave foránea sean válidos y correspondan a valores existentes en la columna de la tabla relacionada (clave primaria).



### Contexto:

Supongamos que tienes dos tablas: "Clientes" y "Pedidos".

La tabla "Pedidos" tiene una clave foránea "id\_cliente" que se relaciona con la clave primaria "id\_cliente" en la tabla "Clientes".





# Insertar datos con integridad referencial



**Paso 1:** Inserta un cliente en la tabla "Clientes" primero:

```
INSERT INTO Clientes (id_cliente, nombre)
VALUES (1, 'Cliente Ejemplo');
```



**Paso 2:** Insertar un pedido en la tabla "Pedidos" asegurándote de que la clave foránea "id\_cliente" se refiera a un cliente existente en la tabla "Clientes":

```
INSERT INTO Pedidos (id_pedido, id_cliente, fecha_pedido)
VALUES (1, 1, '2023-08-17');
```



# Insertar datos con integridad referencial

## Analicemos el ejemplo anterior:

- ✕ En este ejemplo, el valor 1 en la columna "id\_cliente" de la tabla "Pedidos" corresponde al "id\_cliente" insertado en la tabla "Clientes" en el paso anterior.

Esto mantiene la integridad referencial, ya que estás insertando un pedido relacionado con un cliente válido.

Si intentas insertar un pedido con un "id\_cliente" que no existe en la tabla "Clientes", la base de datos arrojará un error de integridad referencial.

Es importante asegurarse de que los valores en las columnas con clave foránea sean válidos antes de realizar la inserción.

Siempre debes insertar primero los registros en la tabla principal (en este caso, "Clientes") y luego insertar registros en las tablas relacionadas (en este caso, "Pedidos") utilizando los valores correspondientes.



# LIVE CODING

Ejemplo en vivo

## Insertión de datos con Integridad referencial | Parte 1:

*En este ejercicio, aprenderás a insertar datos en dos tablas relacionadas: "Usuarios" y "Transacción".*

- La tabla "Transacción" tiene una **clave foránea** "sender\_user\_id" que se relaciona con la **clave primaria** "user\_id" en la tabla "Usuarios".
- La tabla "Transacción" tiene una **clave foránea** "receiver\_user\_id" que se relaciona con la **clave primaria** "user\_id" en la tabla "Usuarios".

*Aseguraremos la integridad referencial insertando primero usuarios y luego transacciones relacionadas con esos usuarios.*

**Tiempo: 10 minutos**



# LIVE CODING

Ejemplo en vivo

## Inserción de datos con integridad referencial | Parte 2:

1. *Insertar al menos tres transacciones en la tabla "Transacciones". Asegúrate de que cada transacción esté relacionada con uno de los usuarios que insertaste en el paso anterior.*
2. *Insertar al menos tres monedas en la tabla "Moneda".*
3. *Escribir una consulta para recuperar y mostrar todas las transacciones junto con el id del usuario al que pertenecen.*

**Tiempo:** 10 minutos

# ➤ Actualizar datos con integridad referencial




# Actualizar datos con integridad referencial




## ¿Cómo actualizar datos con integridad referencial?:

Actualizar datos con integridad referencial en SQL involucra modificar registros de tal manera que se mantengan las relaciones y restricciones definidas entre tablas. Aquí tienes un ejemplo de cómo puedes hacerlo:



### Contexto:

Supongamos que tienes dos tablas: "Autores" y "Libros". La tabla "Libros" tiene una clave foránea "id\_autor" que se relaciona con la clave primaria "id\_autor" en la tabla "Autores".





# Actualizar datos con integridad referencial



**Paso 1:** Actualiza los datos del autor en la tabla "Autores":

```
UPDATE Autores  
SET nombre_autor = 'Nuevo Nombre'  
WHERE id_autor = 1;
```

**Paso 2:** actualiza los datos del libro relacionado con el autor actualizado:

```
UPDATE Libros  
SET titulo_libro = 'Nuevo Título'  
WHERE id_autor = 1;
```

# Actualizar datos con integridad referencial



## **Analicemos el ejemplo anterior:**

En este ejemplo, primero actualizas el nombre de un autor en la tabla "Autores". Luego, actualizas el título de un libro en la tabla "Libros" relacionado con ese autor.

*Asegúrate de que la restricción de clave foránea se mantenga al actualizar datos. La base de datos mantendrá la coherencia y evitará que actualices datos de tal manera que los registros relacionados sean inconsistentes.*



# LIVE CODING

Ejemplo en vivo

## Actualización de datos con integridad referencial | Parte 1:

*En este ejercicio, practicarás cómo actualizar datos en dos tablas relacionadas: "**Transacción**" y "**Usuarios**".*

*La tabla "transacción" tiene una clave foránea "isender\_user\_id" que se relaciona con la clave primaria "user\_id" en la tabla "Usuario".*

**Tiempo: 10 minutos**



# LIVE CODING

Ejemplo en vivo

## Actualización de Datos con Integridad Referencial | Parte 2:

- 1. Recuperar todos los cursos de la tabla "Usuarios" junto con el nombre del usuario al que pertenecen.*
- 2. Actualizar el nombre de un usuario en la tabla "Usuario".*
- 3. Escribir una consulta final para verificar que los datos se hayan actualizado correctamente y que la relación entre "Usuario" y "Transacción" se mantenga intacta.*

**Tiempo:** 10 minutos

# ➤ Eliminar datos con integridad referencial




# Eliminar datos con integridad referencial




## ¿Cómo eliminar datos con integridad referencial?:

Eliminar datos con integridad referencial en SQL implica asegurarte de que las eliminaciones no rompan las relaciones y restricciones definidas entre las tablas.



### Contexto:

Supongamos que tienes dos tablas: "Autores" y "Libros". La tabla "Libros" tiene una clave foránea "id\_autor" que se relaciona con la clave primaria "id\_autor" en la tabla "Autores".





# Eliminar datos con integridad referencial



**Paso 1:** Eliminar un libro en la tabla "Libros" asegurándote de que la integridad referencial se mantenga:

```
DELETE FROM Libros  
WHERE id_libro = 1;
```



**Paso 2:** Eliminar al autor del libro que acabas de eliminar:

```
DELETE FROM Autores  
WHERE id_autor = 1;
```

# Eliminar datos con integridad referencial



## **Analicemos el ejemplo anterior:**

En este ejemplo, primero eliminas un libro de la tabla "Libros" y luego, eliminando al autor asociado en la tabla "Autores".

La base de datos mantendrá la coherencia y evitará eliminar registros que rompan relaciones.

*Recuerda que al trabajar con integridad referencial, debes ser cuidadoso al eliminar registros en tablas relacionadas.*






# Eliminar datos con integridad referencial




## ¿Cómo eliminar datos en cascada?:

La opción "**CASCADE**" en las restricciones de clave foránea permite que las operaciones de eliminación o actualización en la tabla principal (la tabla referenciada por la clave foránea) se propaguen automáticamente a las tablas secundarias (las tablas que tienen la clave foránea).



Esto significa que puedes eliminar registros en la tabla principal y las filas relacionadas en las tablas secundarias también se eliminarán automáticamente.



# Eliminar datos con integridad referencial

## ¿Cómo eliminar datos en cascada?:

- ✕ Para eliminar datos en cascada simplemente debes implementar la opción "ON DELETE CASCADE" al crear la tabla. Aquí tienes un ejemplo

### Paso 1:

```
CREATE TABLE Libros (  
  id_libro INT PRIMARY KEY,  
  titulo_libro VARCHAR(100),  
  id_autor INT,  
  FOREIGN KEY (id_autor)  
  REFERENCES Autores(id_autor) ON DELETE CASCADE  
);
```

### Paso 2:

```
DELETE FROM Autores WHERE  
id_autor = 1;
```

- En este ejemplo, cuando se elimina el autor con "id\_autor" 1 de la tabla "Autores", el libro relacionado en la tabla "Libros" también se elimina automáticamente debido a la opción "ON DELETE CASCADE" en la
- ✕ restricción de clave foránea.

# Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro...

## Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase.





# LIVE CODING

Ejemplo en vivo

## Insertión y eliminación de Datos con Integridad Referencial - Clientes y Pedidos

1. *Crear dos tablas: "Clientes" y "Pedidos". Asegurarse de que la tabla Pedidos tenga al menos: id\_pedido, id\_cliente (Clave foránea que se relaciona con "id\_cliente" en la tabla "Clientes" con ON DELETE CASCADE).*
2. *Eliminar uno de los clientes de la tabla "Clientes". Asegúrate de que los pedidos relacionados con ese cliente también se eliminen automáticamente debido a la restricción "ON DELETE CASCADE".*

**Tiempo:** 10 minutos



# LIVE CODING

Ejemplo en vivo

## Eliminar de datos con integridad referencial

*En este ejercicio, practicarás cómo eliminar datos en dos tablas relacionadas: "Transacción" y "Usuarios". La tabla "transacción" tiene una clave foránea "sender\_user\_id" que se relaciona con la clave primaria "user\_id" en la tabla "Usuario".*

*Aseguraremos que la eliminación de un usuario también elimine automáticamente las transacciones relacionadas.*

**Tiempo:** 10 minutos

○

# ¿Alguna consulta?

+



# RESUMEN

¿Qué logramos en esta clase?

- ✓ **Conocer el concepto de Integridad referencial**
- ✓ **Aprender a insertar datos con integridad referencial**
- ✓ **Aprender a actualizar datos con integridad referencial**
- ✓ **Aprender a eliminar datos con integridad referencial**



# #WorkingTime

Continuemos ejercitando

**¡Antes de cerrar la clase!** Te invitamos a: 📌 📌 📌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
  - a. Material 1 (*Lectura de la Lección 3: Sentencias para la manipulación de datos y transaccionalidad, páginas 9-12*)
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

# ¡Muchas Gracias!

Nos vemos en la próxima clase 🙌



Momento: ✚

# Time-out!

🕒 5 min.

