



# Sistemas de Bases de Datos 2

Diego André Mazariegos Barrientos

<b>Día, Fecha:</b>	Miércoles 31 de julio de 2024
<b>Hora de inicio:</b>	17:20

# CLASE 2

## TRANSACCIONES

### PARTE 1

SISTEMAS DE BASES DE DATOS 2

# AGENDA

- Repaso clase anterior
  - DML, DDL, DCL
  - Relaciones recursivas, (supertipo y subtipo) y arcos
- Repositorio del curso
- Recordatorio foros
- DBMS Lab y Ranking
- Transaccionalidad
  - Introducción
  - Conceptos allegados
  - ACID
  - Tipos de transacciones
- Concurrencia
  - Problemas de concurrencia
  - Técnicas de control
- PL/SQL
  - Introducción
  - SP's y PL/SQL
- TSQL
  - Transacciones y TSQL
- Ejemplos prácticos

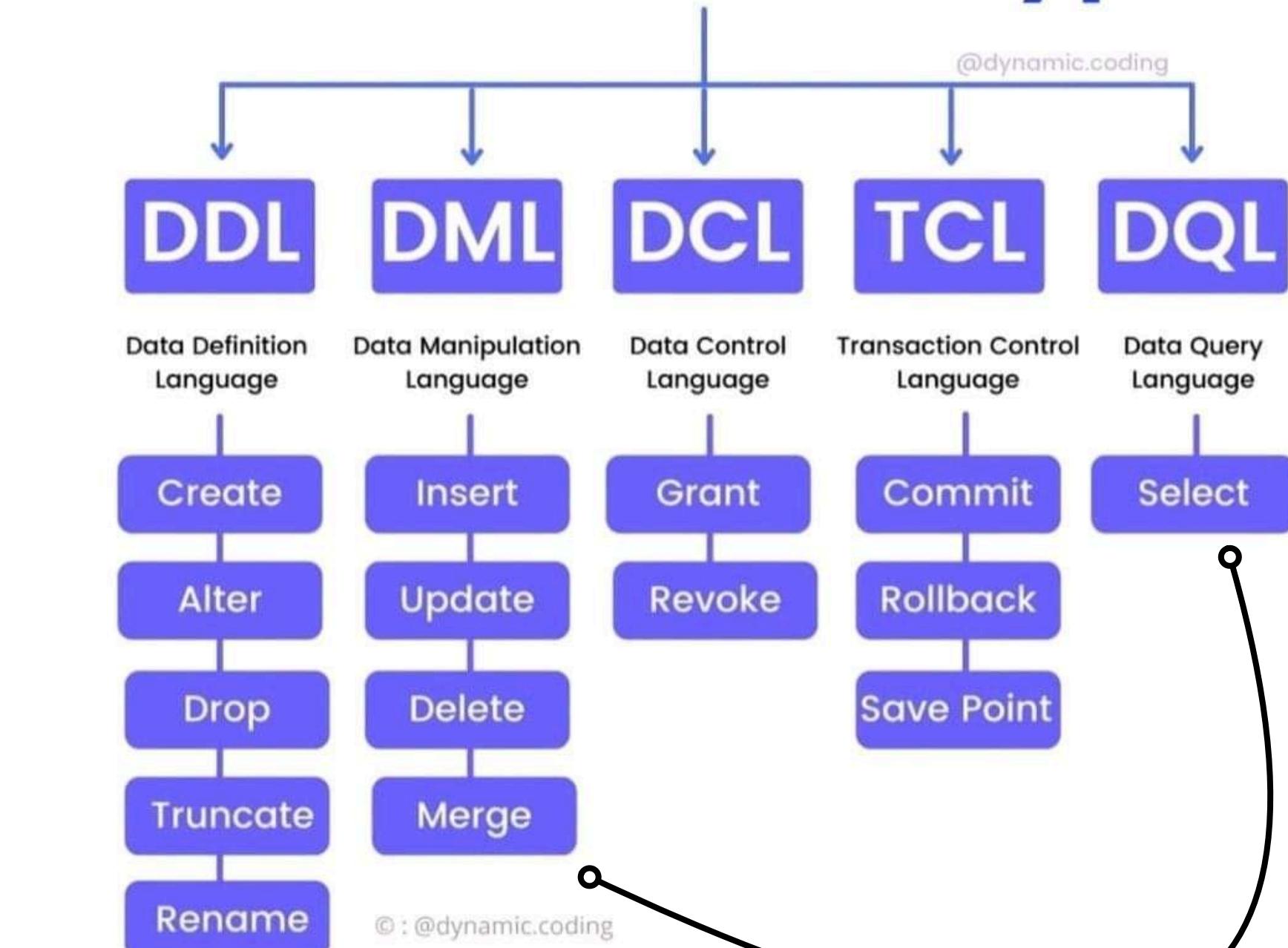
# Continuación Repaso

# DML, DCL, DDL y nuevo concepto TCL, DQL

## NOTA

Perspectivas **SELECT**: En la mayoría de los contextos y materiales de enseñanza, **SELECT** se incluye dentro de **DML** por su papel en la manipulación y consulta de datos. Sin embargo, también es correcto referirse a **SELECT** como parte de **DQL** para enfatizar su función específica de consulta.

# SQL Command Types



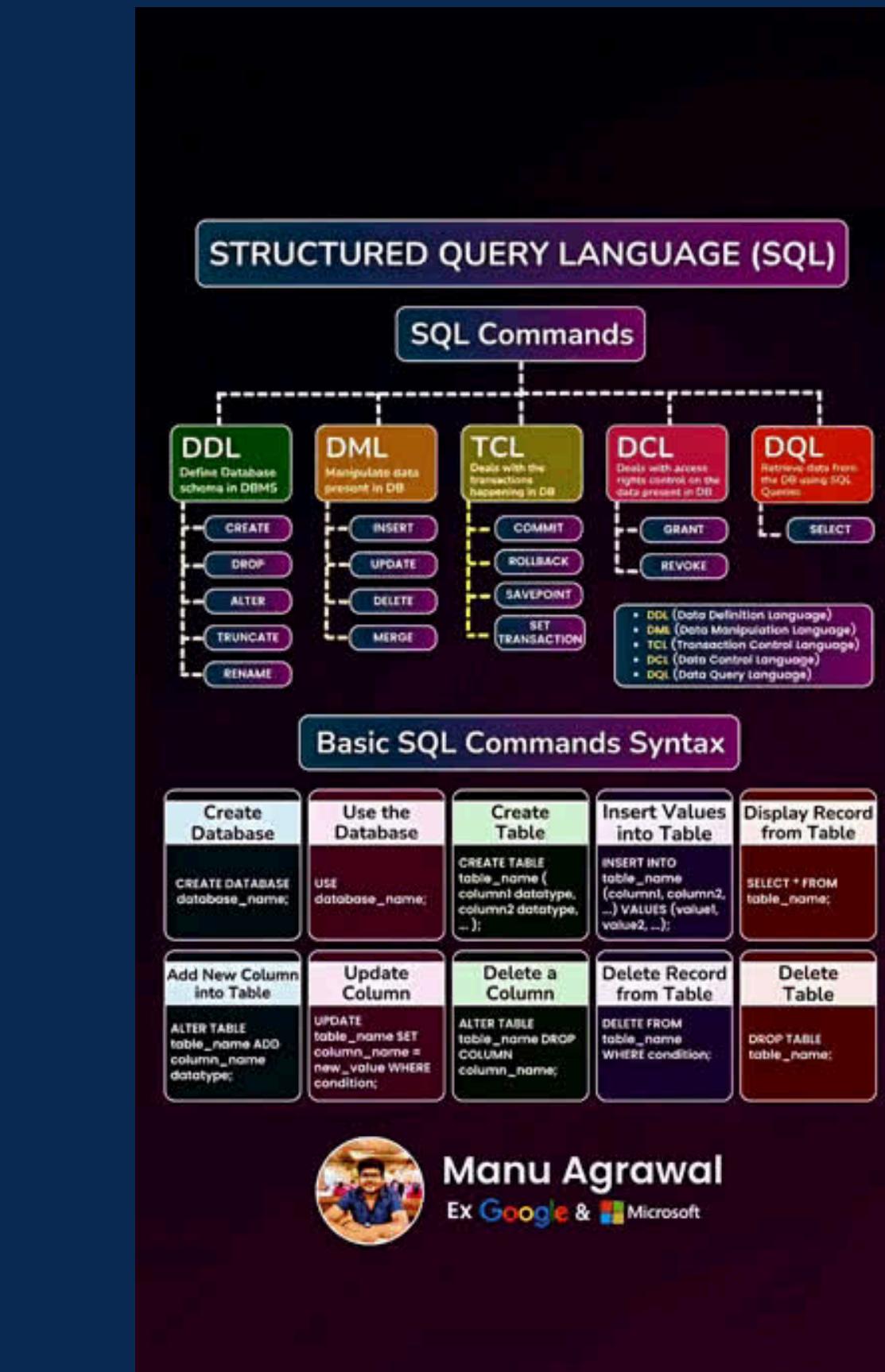
Controversia

# DML, DCL, DDL y nuevo concepto TCL, DQL

```
5   + Prevent database truncation if the table has no primary key
6   abort("The Rails environment is running in production mode!")
7   require 'spec_helper'
8
9   require 'capybara/rspec'
10  require 'rspec/rails'
11
12  # Capybara.javascript_driver = :webkit
13  # Category.delete_all; Category.create!(name: "Electronics")
14  # Shoulda::Matchers.configure { |config| config.integrate_with FactoryGirl }
15  # FactoryGirl.libraries_dir_name = 'factories'
16  # FactoryGirl.find_definitions
17  # FactoryGirl.define do
18  #   factory :category do
19  #     name "Electronics"
20  #   end
21  # end
22  # Add additional requirements here
23  # Requires supporting files
24  # spec/support/ and its
25  # run as spec files by
26  # in_spec.rb will both be
27  # run twice. It is recom-
28  # mended to not include
#       no results found for 'mongoid'
```

## NOTA

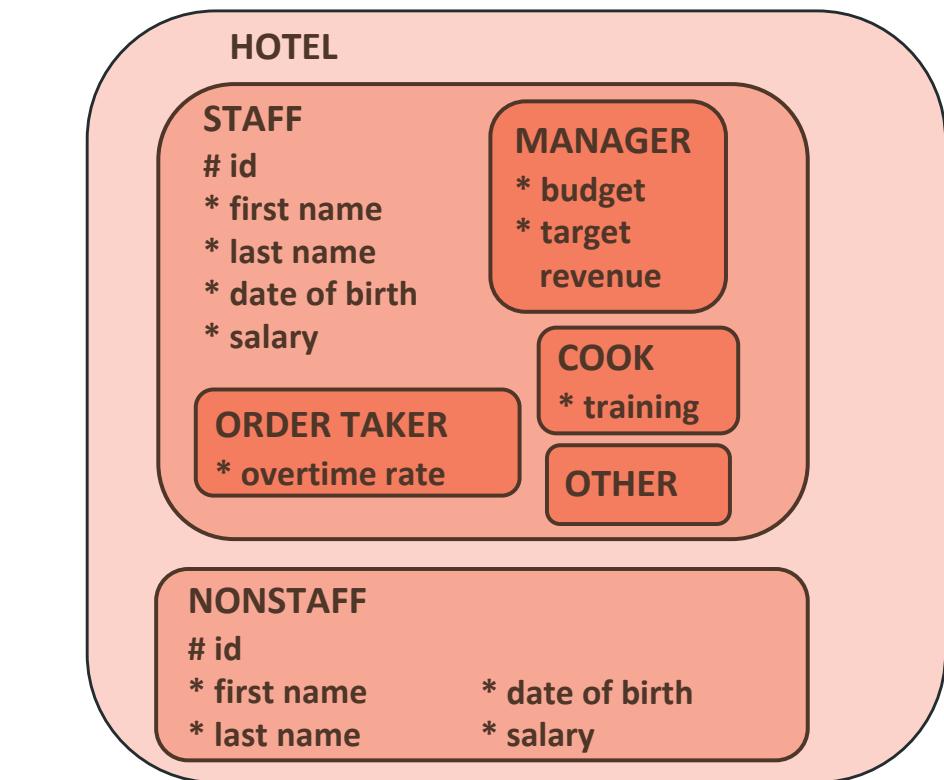
Perspectivas **SELECT**: En la mayoría de los contextos y materiales de enseñanza, **SELECT** se incluye dentro de **DML** por su papel en la manipulación y consulta de datos. Sin embargo, también es correcto referirse a **SELECT** como parte de **DQL** para enfatizar su función específica de consulta.



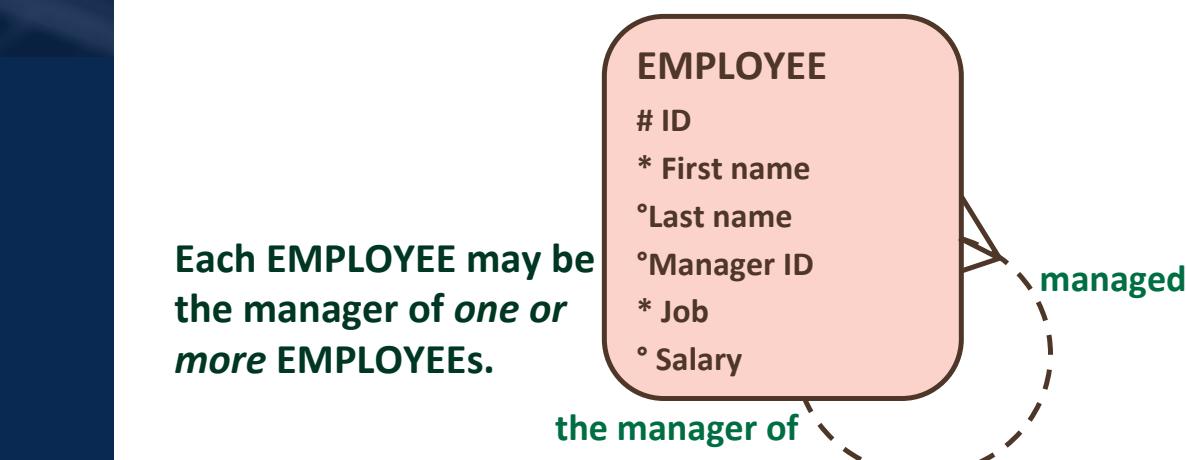
# Relaciones comunes en SQL

```
5   + Prevent database truncation if no migration is present
6   abort("The Rails environment is missing a database configuration")
7   require 'spec_helper'
8   require 'rspec/rails'
9
10  require 'capybara/rspec'
11  require 'capybara/rails'
12
13  Capybara.javascript_driver = :webkit
14  Shoulda::Matchers.configure do |config|
15    config.integrate do |spock|
16      spock.with_test_framework :rspec
17      spock.with_library :rails
18    end
19  end
20
21  # Add additional requirements here
22
23  # Requires supporting files within the same directory as the spec file or explicitly
24  # spec/support/ and its subdirectories to be available.
25  # run as spec/ and its subdirectories to be available.
26  # in _spec.rb will both be available.
27  # run twice. It is recommended to keep it in the top level spec directory so it runs
28  # automatically on the command line. You can specify the :files option to run
# additional files in the spec directory.
# no results found for 'mongoid'
```

## Supertipo STAFF anidado

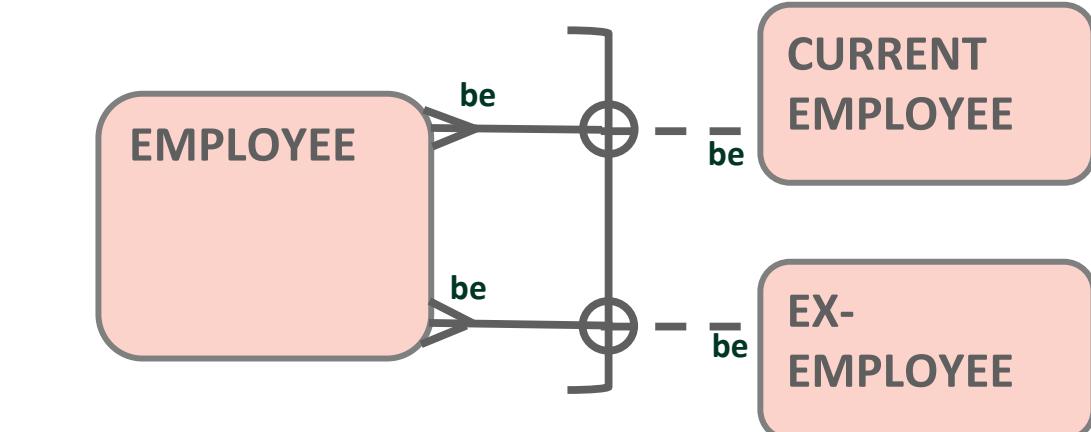


## Recursive Relationships



Each **EMPLOYEE** may be the manager of *one or more* **EMPLOYEE**s.

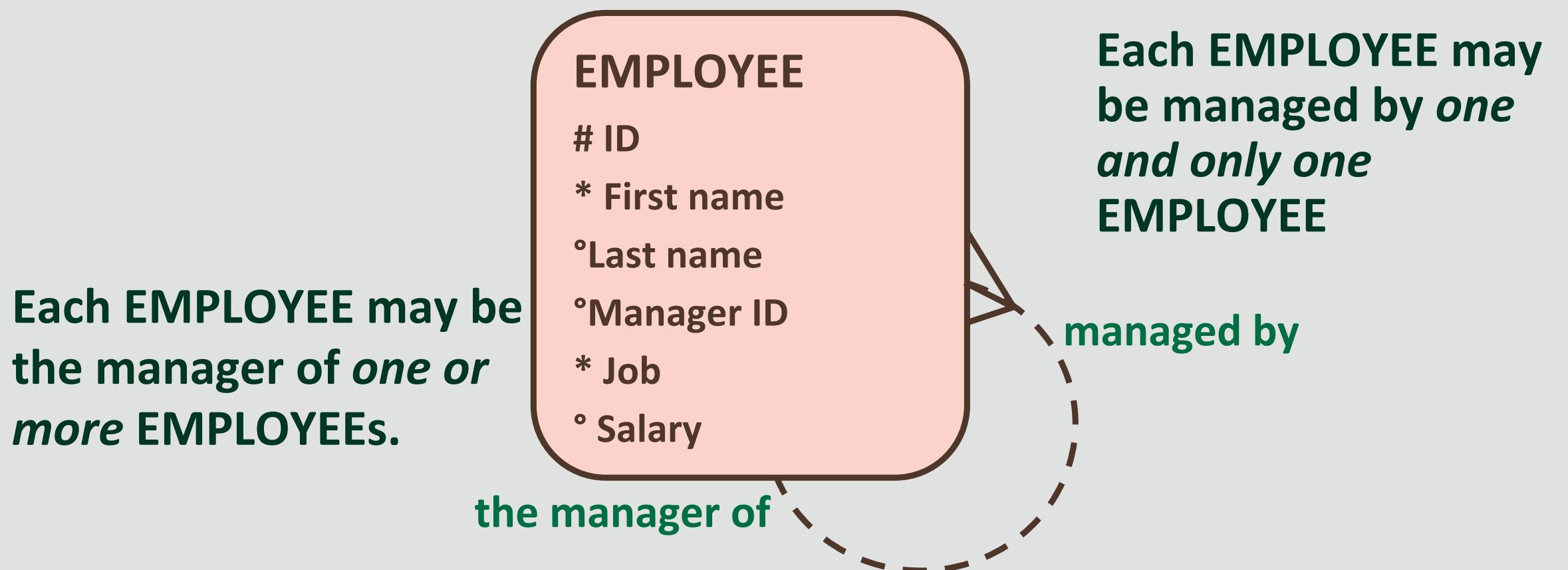
## Arc Relationships



# Relaciones recursivas

- Una relación recursiva es aquella en la que una instancia de entidad está relacionada con otra instancia de la misma entidad.

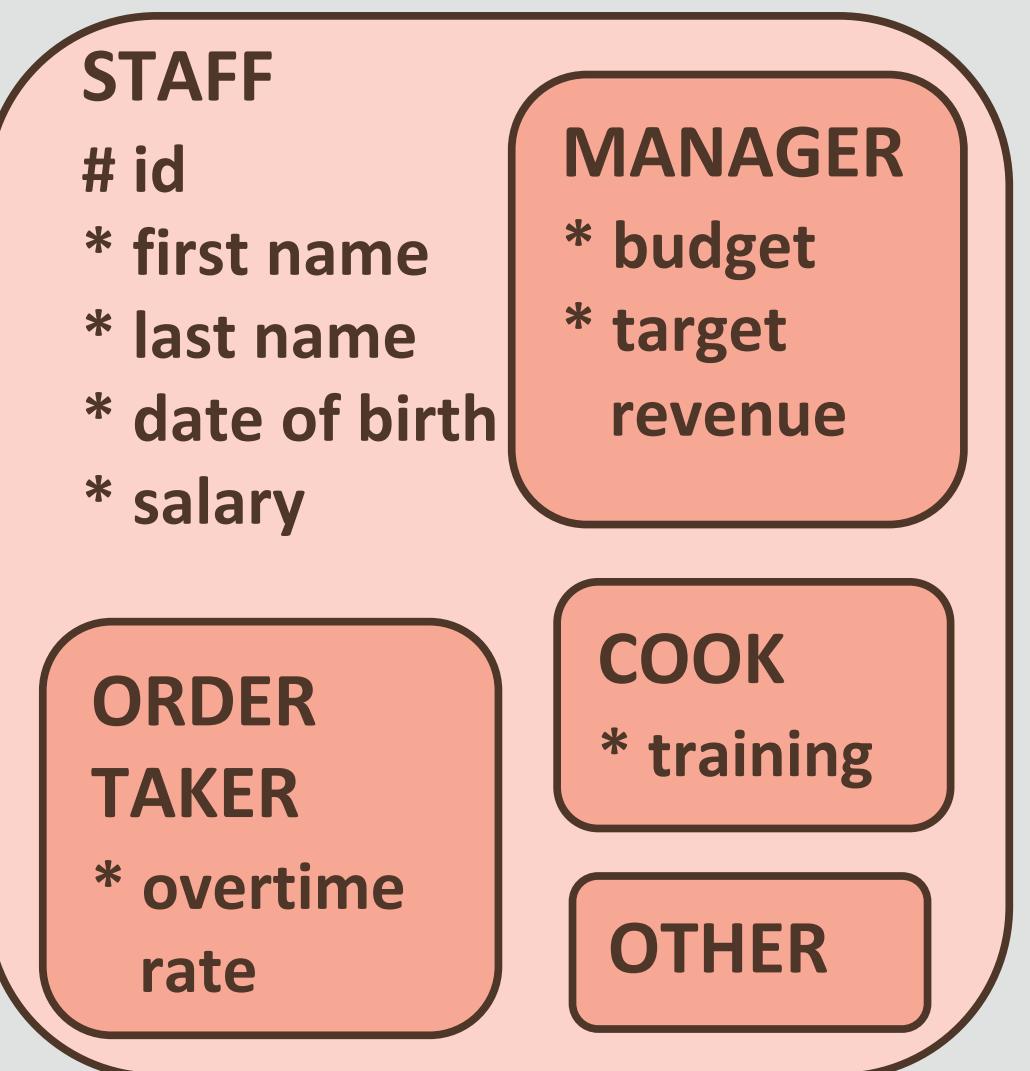
- En este ejemplo, cada gerente es también un empleado. Por lo tanto, gerente no es una entidad nueva, sino solo un subconjunto de las instancias de la entidad EMPLEADO. Esta relación de varios a uno es opcional en ambas direcciones. Es lo mismo que la relación entre dos entidades distintas.



# Supertypes and subtypes (nested)

El subtipo es una subagrupación de la entidad en un tipo de entidad que tiene atributos que son distintos de los de otras subagrupaciones (no necesariamente excluyentes).

- Hereda todos los atributos del supertipo.
- Hereda todas las relaciones del supertipo.
- Normalmente tiene sus propios atributos o relaciones.
- Se dibuja dentro del supertipo.
- Nunca existe solo.
- Puede tener sus propios subtipos.
- Tiene claves primarias idénticas del supertipo y el subtipo

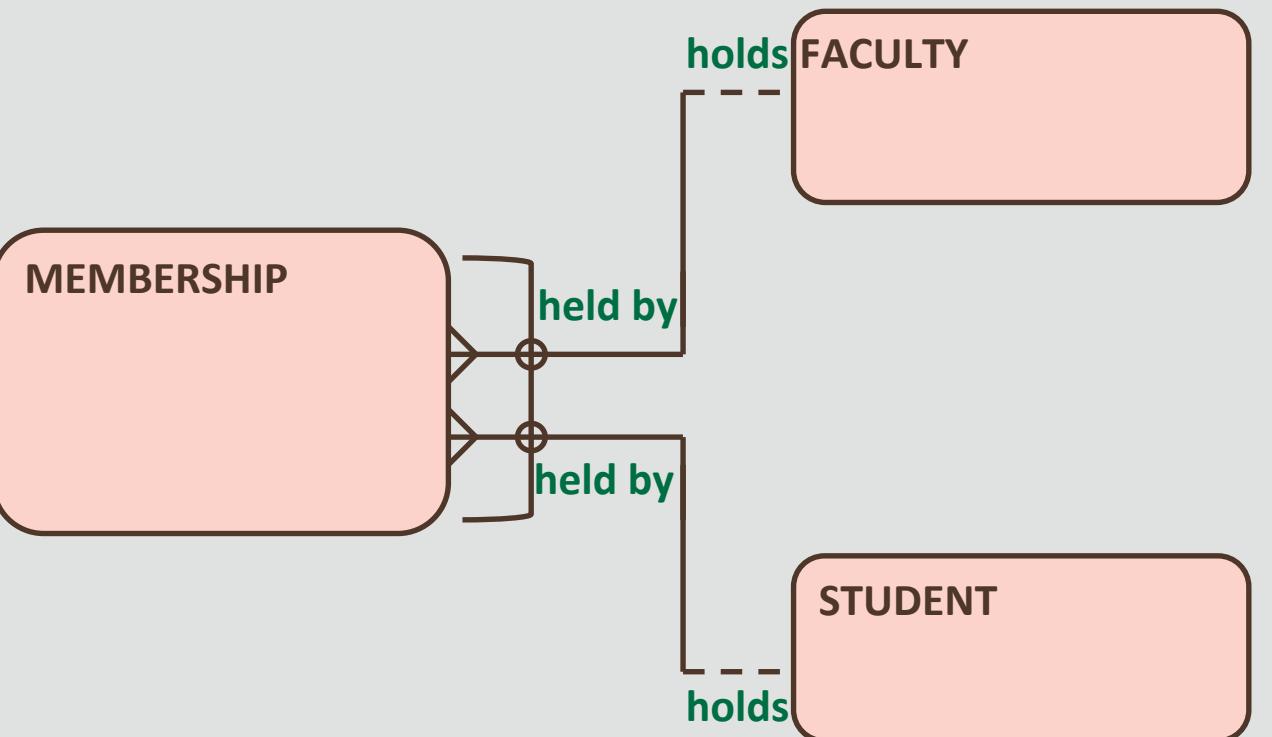


- **Exhaustivo**
  - Cada instancia de un supertipo es también una instancia de uno de los subtipos.
  - Otro debe incluirse como subtipo para categorizar instancias que no están definidas por uno de los subtipos existentes.
  - Ejemplo: un empleado debe ser de tiempo completo, tiempo parcial u otro.
- **Mutuamente excluyentes**
  - Cada instancia del supertipo es de un solo subtipo.
  - Ejemplo: Un empleado no puede trabajar a tiempo completo y a tiempo parcial.

# Arc Relationships

Una relación arqueada es un grupo de relaciones mutuamente excluyentes para una entidad. Esto significa que una instancia de la entidad solo puede participar en una de las relaciones del grupo a la vez.

- Exclusividad: Solo una de las relaciones en el arco puede existir para una instancia de la entidad en cualquier momento.
- Representación Visual: Se representa con una línea en forma de arco que conecta las relaciones, mostrando la exclusividad mutua.
- Cardinalidad: Todas las relaciones en el arco deben tener la misma cardinalidad (todas opcionales o todas obligatorias).
- Entidad Única: El arco pertenece a una sola entidad y solo incluye relaciones que se originan de esa entidad.

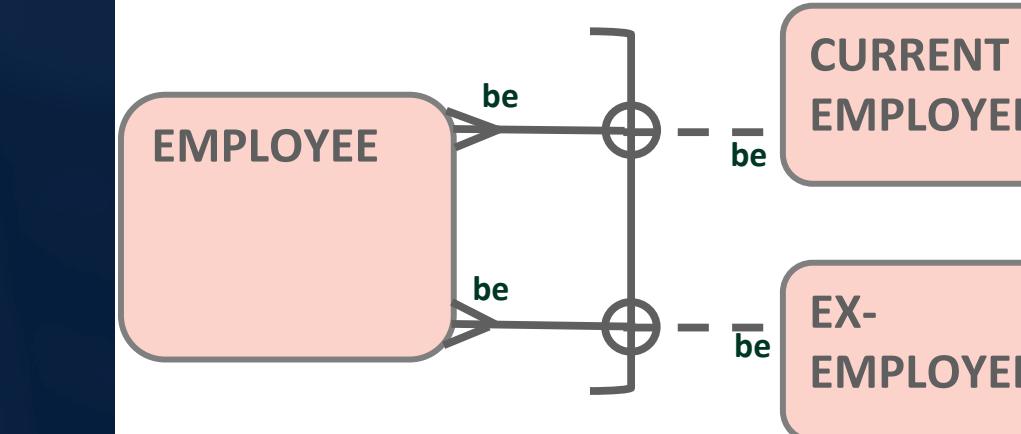


- Cada MEMBRESÍA puede ser titular de un único FACULTAD o puede ser titular de un único ESTUDIANTE.
- Cada FACULTAD puede ser titular de una o más MEMBRESÍAS. Cada ESTUDIANTE puede ser titular de una o más MEMBRESÍAS.
- Una MEMBRESÍA no puede ser titular tanto de un FACULTAD como de un ESTUDIANTE. Esta exclusividad mutua está modelada por el arco.
- Se pueden dibujar arcos para relacionesopcionales y para relaciones obligatorias, pero las dos relaciones que participan en el arco deben tener la misma cardinalidad.

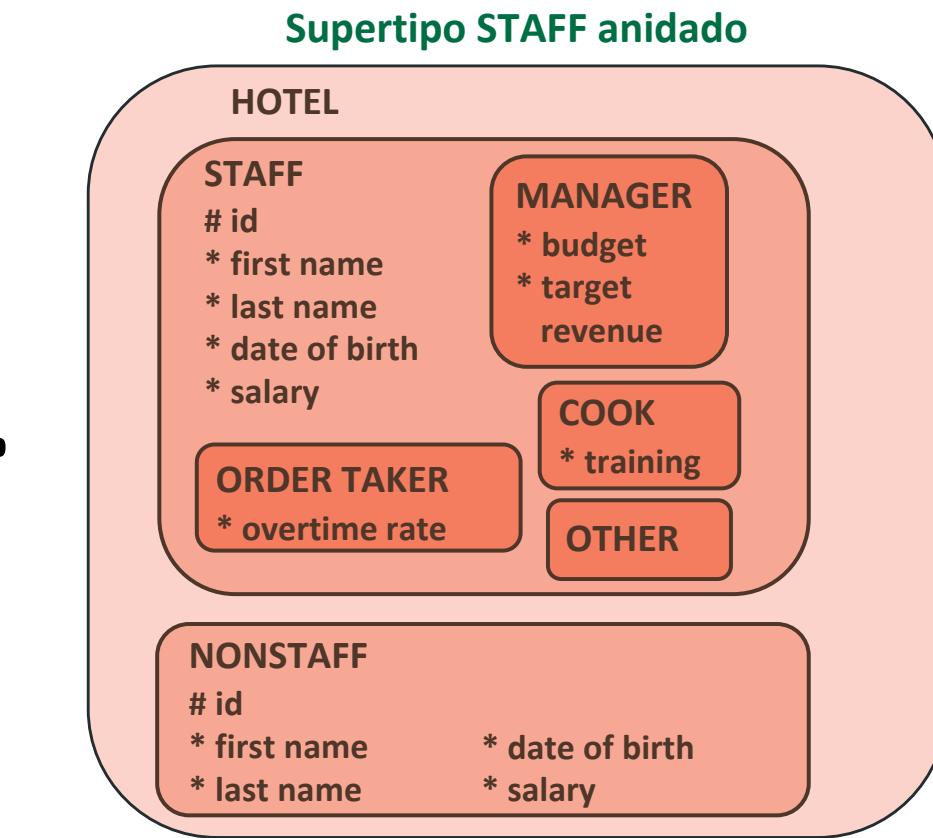
# Relaciones comunes en SQL

```
5   abort("The Rails environment is running in production mode")
6   require 'spec_helper'
7   require 'rspec/rails'
8
9   require 'capybara/rspec'
10  require 'capybara/rails'
11
12  Capybara.javascript_driver = :webkit
13  Category.delete_all; Category.create!
14  Shoulda::Matchers.configure do |config|
15    config.integrate_with :rspec
16    config.integrate_with :shoulda_matchers
17    config.integrate_with :expectations
18    config.integrate_with :declarative_matchers
19  end
20
21  # Add additional requirements here
22  # Requires supporting files
23  # spec/support/ and its
24  # run as spec files by
25  # in_spec.rb will both be
26  # run twice. It is recom-
27  # mended to put the
28  # run twice. It is recom-
# in_spec.rb will both be
# run twice. It is recom-
# mended to put the
# run twice. It is recom-
```

## Arc Relationships



VS.



## Naturaleza de la Relación

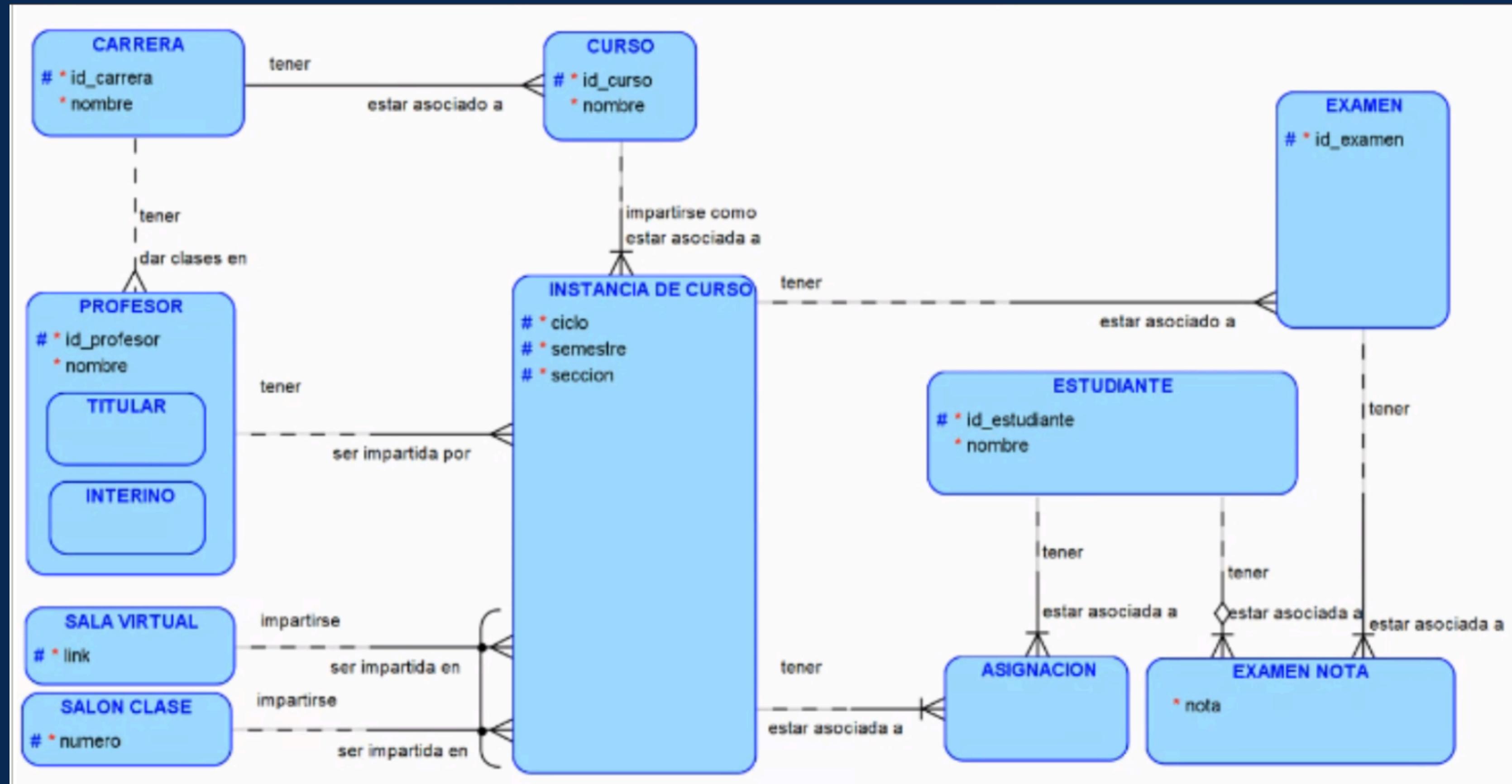
- **Supertiros/Subtipos:** Involucran herencia y jjerarquía donde los subtipos son especializaciones del supertipo.
- **Arcos:** Involucran exclusividad/catalogan entre relaciones alternativas para una entidad única, indicando que solo una de las relaciones puede aplicarse a la vez.

## Aplicación en Modelos de Datos

- **Supertiros/Subtipos:** Utilizado para modelar entidades que comparten características comunes pero también tienen atributos específicos.
- **Arcos:** Utilizado para modelar escenarios donde una entidad tiene múltiples relaciones potenciales pero solo una puede ser verdadera en cualquier momento.

D u d a s ?

# Esquema base de datos laboratorio





# Repositorio del laboratorio

[diegomazariegos2002/  
\*\*GUIDE\\_TO\\_DATABASES\\_...\*\*](#)



👤 1  
Contributor

⌚ 0  
Issues

⭐ 0  
Stars

🍴 0  
Forks

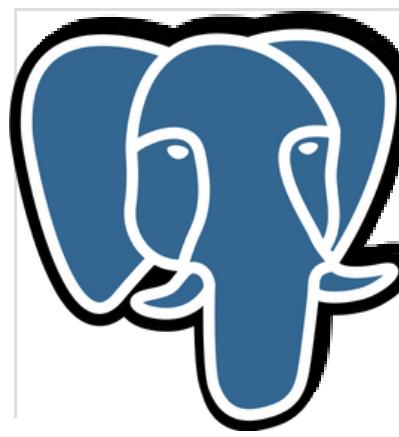


**diegomazariegos2002/GUIDE\_TO\_DATABASES\_BD2\_2S24**

Contribute to diegomazariegos2002/GUIDE\_TO\_DATABASES\_BD2\_2S24 development by creating an account on GitHub.

GitHub

# FORO SEMANA # 2



## PostgreSQL 16.3 Documentation

PostgreSQL 16.3 Documentation The PostgreSQL Global Development Group Copyright © 1996–2024 The PostgreSQL Global Development Group Legal Notice Tab...

 PostgreSQL Documentation / May 9

## ¿Qué DBMS se utilizará?

<https://db-engines.com/en/>

### Pros

- Open Source: es gratuito y de código abierto, con una comunidad activa.
- Compatibilidad multiplataforma: Funciona bien en varios sistemas operativos como Windows, Linux y macOS.
- Características avanzadas: Soporta características avanzadas como transacciones complejas, funciones, y procedimientos almacenados.
- Extensibilidad: permite la creación de extensiones y tiene capacidades avanzadas de manejo de datos geoespaciales (PostGIS).

### Contras:

- Curva de aprendizaje: Puede tener una curva de aprendizaje más pronunciada debido a sus características avanzadas.
- Soporte: Aunque la comunidad es activa, el soporte comercial no es tan extenso como en otros DBMS.

# TRANSACCIONES

# Transacciones

## Definición

Una transacción es una unidad de trabajo que se ejecuta de manera completa o no se ejecuta en absoluto. En términos de bases de datos, una transacción es una secuencia de operaciones SQL que se ejecutan como una sola unidad lógica.

## Propósito

Asegurar la integridad de los datos y permitir que las operaciones se realicen de manera segura y coherente.

```
1 -- EJEMPLO 1
2 BEGIN;
3 UPDATE cuentas SET saldo = saldo - 100 WHERE cuenta_id = 1;
4 UPDATE cuentas SET saldo = saldo + 100 WHERE cuenta_id = 2;
5 COMMIT;
6
7 -- EJEMPLO 2
8 BEGIN;
9 UPDATE vuelos SET asientos_disponibles = asientos_disponibles - 1 WHERE vuelo_id = 123;
10 INSERT INTO reservas (cliente_id, vuelo_id) VALUES (456, 123);
11 COMMIT;
```

# ACID

Atomicity  
Consistency  
Isolation  
Durability

# Atomicidad

- Definición: La atomicidad asegura que todas las operaciones en una transacción se completen exitosamente. Si alguna operación falla, toda la transacción se revierte.
- Ejemplo: Si la transferencia de dinero falla en una de las operaciones, ninguna de las operaciones tendrá efecto.

```
15 -- ATOMICIDAD
16
17 BEGIN;
18 UPDATE cuentas SET saldo = saldo - 100 WHERE cuenta_id = 1;
19 UPDATE cuentas SET saldo = saldo + 999 WHERE cuenta_id = 2;
20 -- Simulación de un error
21 ROLLBACK;
```

ACID

# Consistencia

- Definición: La consistencia garantiza que una transacción lleve la base de datos de un estado válido a otro estado válido, manteniendo las reglas y restricciones de la base de datos.
- Ejemplo: Las restricciones de claves foráneas deben mantenerse durante una transacción.

```
24 BEGIN;
25 UPDATE cuentas SET saldo = saldo - 100 WHERE cuenta_id = 1;
26 -- Violación de una restricción de clave foránea
27 -- Si el cliente 999 no existe, se revierte.
28 INSERT INTO reservas (cliente_id, vuelo_id) VALUES (999, 123);
29 ROLLBACK;
```

A C I D

# Aislamiento

- Definición: El aislamiento asegura que las transacciones concurrentes se ejecuten como si fueran secuenciales, evitando interferencias.
- Niveles de Aislamiento en PostgreSQL:
  - READ UNCOMMITTED
  - READ COMMITTED (por defecto en PostgreSQL)
  - REPEATABLE READ
  - SERIALIZABLE
- Ejemplo: Demostración del nivel de aislamiento READ COMMITTED.

```
31 -- AISLAMIENTO
32
33 -- Sesión 1
34 BEGIN;
35 UPDATE cuentas SET saldo = saldo - 100 WHERE cuenta_id = 1;
36
37 -- Sesión 2
38 -- No verá el cambio hasta que se haga COMMIT en la sesión 1.
39 SELECT saldo FROM cuentas WHERE cuenta_id = 1;
```

ACID

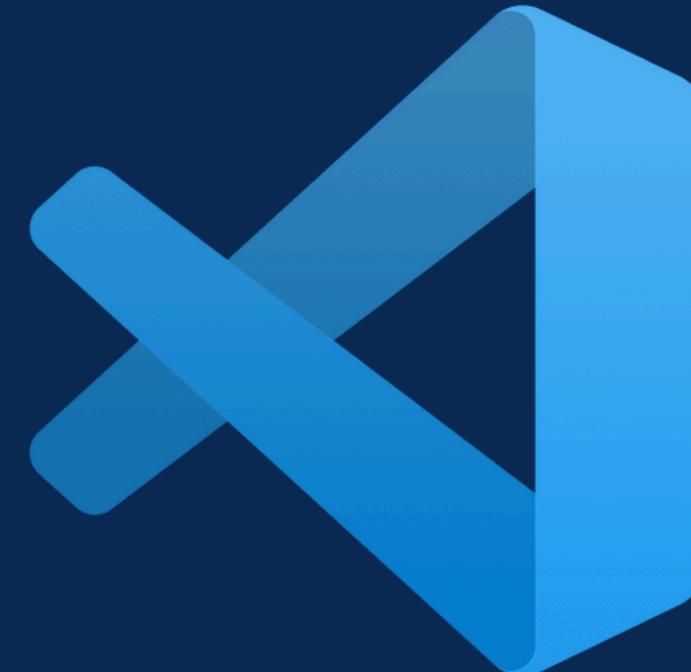
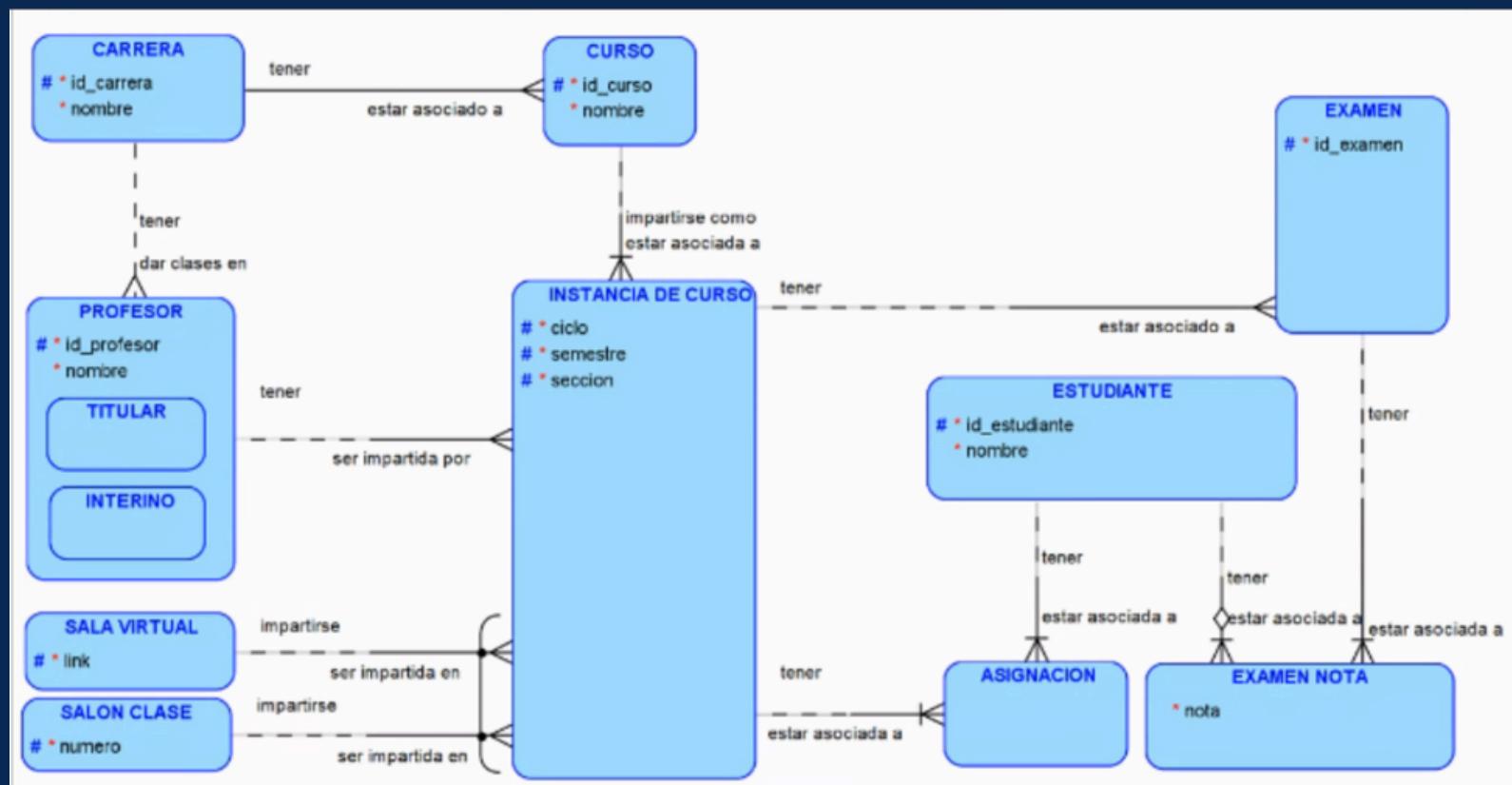
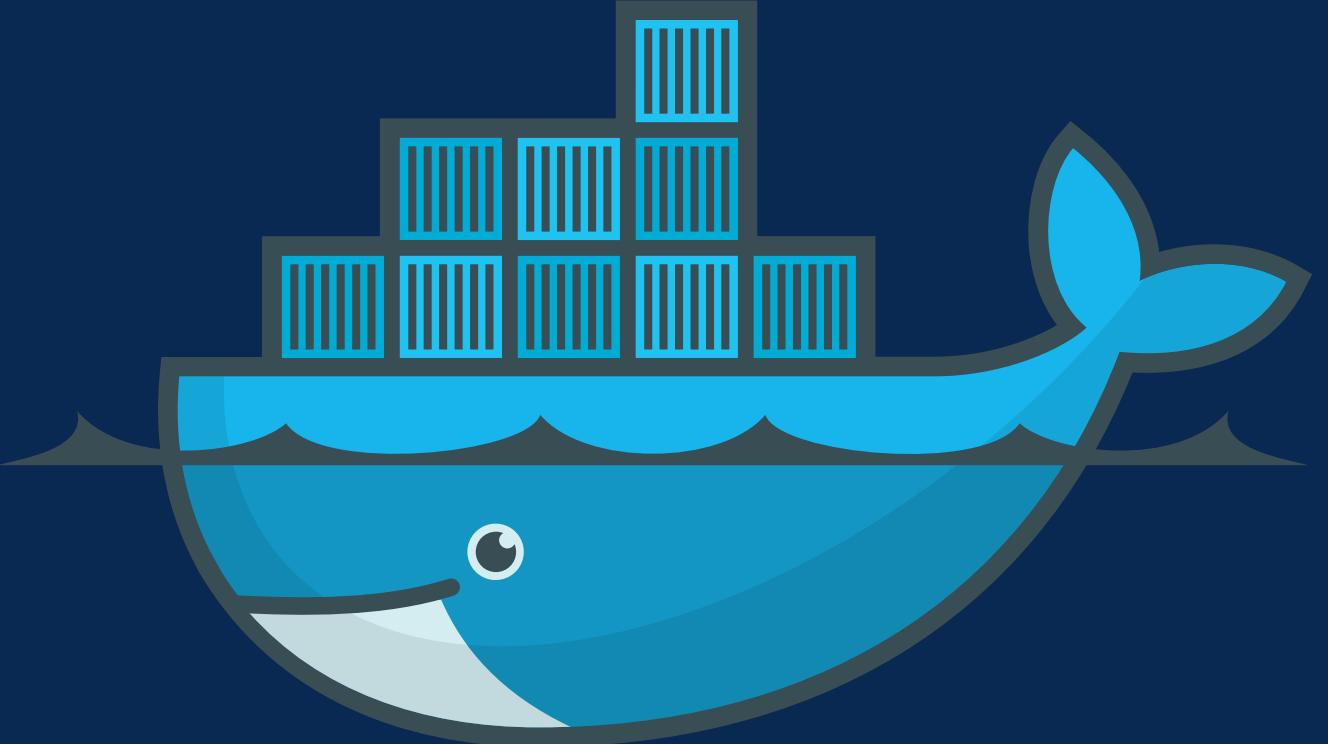
# Durabilidad

- Definición: La durabilidad asegura que una vez que una transacción ha sido confirmada (committed), sus cambios persisten incluso en caso de un fallo del sistema.
- Ejemplo: Los datos escritos durante una transacción confirmada permanecen en la base de datos.

```
41 -- DURABILIDAD
42
43 BEGIN;
44 UPDATE cuentas SET saldo = saldo - 100 WHERE cuenta_id = 1;
45 COMMIT;
46 -- Después de un fallo y recuperación del sistema, el cambio sigue presente.
```

A C I D

# EJEMPLO E INSTALACIÓN DE HERRAMIENTAS



# Por qué generar los IDs desde el Frontend y No desde la Base de Datos: FAQs | #lafunción 9x3



**CodelyTV - Redescubre la programación**

Sube de nivel. Haz código más mantenible, escalable y testable 🎉 Codely enseña y entretiene...

[YouTube](#)

RECOMENDACION DE LA SEMANA

