

# Base de Datos Avanzada



**Ing. Yogledis Herrera**

# Descripción de la Asignatura

La asignatura está orientado a la utilización de **herramientas** que permitan añadir a las bases de datos un mejor rendimiento y aumentar la seguridad, disminuyendo posibles fallas del sistema y garantizando la integridad de los datos, dándole un toque profesional al diseño y modelado, optimizando el rendimiento al trabajar con altos volúmenes de datos al implementar técnicas que agilicen las consultas.

La asignatura iniciará con la elaboración de un **modelado de bases de datos**, luego se generará **diccionario de datos**, **scripts de la base de datos** y **generación de la base de datos a través de herramientas CASE**, se analizará la integridad de datos mediante llaves primarias, únicas, foráneas, restricciones que se pueden realizar por funciones proporcionado por el gestor de base de datos, también trataremos sobre **manejo de transacciones**, desarrollo de **funciones y procedimientos almacenados** que permitan reforzar la integridad de datos, así como también optimizar el rendimiento de las consultas a través de **índices** y por último se estudiará sobre criterios para mejorar la seguridad de las bases de datos a través de la creación de **usuarios y otorgar privilegios, manejo de esquemas y privilegios, y creación de vistas**.

# Importancia

**Base de Datos** es una asignatura de especialidad de la carrera profesional de Tecnología Superior en Desarrollo de Software, en esta asignatura se complementa las competencia de modelamiento de sistemas informáticos, generación de reportes, se mejora el rendimiento y las seguridades de las bases de datos.

**Crear base de datos** **íntegras, seguras, rápidas, disponibles** y con **confiabilidad de los datos** es de suma importancia para las organizaciones.

# Temarios

## Unidad 1/Generación de Base de datos a través de herramientas Case.

- 1.1. Definición de reglas del negocio en un caso de estudio
- 1.2. Elaboración de un modelo lógico normalizado en una herramienta CASE.
- 1.3. Generación del Script de la base de datos a través de una herramienta CASE (pgmodeler).
  - 1.3.1 Generar Diccionario de datos
  - 1.3.2 Generar Script de la base de datos
  - 1.3.3 Importar un modelo a una base de datos
- 1.4. examen Parcial 1

# Temarios

## Unidad 2/Programación de SQL avanzados

### 2.1. Integridad

#### 2.1.1 Transacciones

#### 2.1.2 Funciones

#### 2.1.3 Procedimientos Almacenados

### 2.2 Rendimiento

#### 2.2.1 . Ingreso de datos Masivos

#### 2.2.2 . Índices

### 2.3 Seguridad

#### 2.3.1. Usuarios

#### 2.3.2 Esquemas

#### 2.3.3. Vistas

#### 2.3.4 RespalDOS

### 2.4. examen Parcial 2

# Planificación

**Semana 1. desde el 01 de Febrero hasta 05 de febrero del 2021**

## **Unidad 1/Generación de Base de datos a través de herramientas Case.**

- 1.1. Definición de reglas del negocio en un caso de estudio
- 1.2. Elaboración de un modelo lógico normalizado en una herramienta CASE.
- 1.3. Generación del Script de la base de datos a través de una herramienta CASE (pgmodeler).
  - 1.3.1 Generar Diccionario de datos
  - 1.3.2 Generar Script de la base de datos
  - 1.3.3 Importar un modelo a una base de datos
- Esquemas
- 1.4. examen Parcial 1

# Planificación

## Semana 2. desde el 08 de Febrero hasta 12 de febrero del 2021

### 2.1. Integridad

#### 2.1.1 Vistas

#### 2.1.2 Funciones

#### 2.1.3 Procedimientos Almacenados

### 2.2 Rendimiento

#### 2.2.1 . Ingreso de datos Masivos

#### 2.2.2 . Índices

# Planificación

**Semana 3. desde el 15 de Febrero hasta 19 de febrero del 2021**

2.1.1 Transacciones

2.3 Seguridad

2.3.1. Usuarios

2.3.2 Esquemas

2.3.3. Vistas

2.3.4 Respaldos



# Planificación

**Semana 4. desde el 22 de Febrero hasta 26 de febrero del 2021**

2.1.1 Desarrollo y Evaluación de Proyecto

# Forma de Evaluar

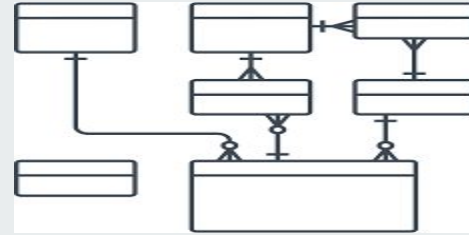
**VER EL EVA**

# Evaluación Diagnostica

**VER EL EVA**

# Clase 1

## Presentación Práctica Herramienta CASE pgmodeler



# Actividades

**Repaso: Modelo Relacional**

**Repaso: Construcción del Modelo Relacional**

**ver video: Realizar un diseño con la herramienta pgmodeler (ejercicio estudiante-estado civil)**

**<https://youtu.be/9rs5uKegvJ0>**

**ver video: Generar diccionario de datos con pgmodeler**

**<https://youtu.be/I2E-CZ7Mn2w>**

**ver video: Exportar un modelo a un Script de la base de dato**

**<https://youtu.be/PIJlkD-ulDI>**

## Actividades

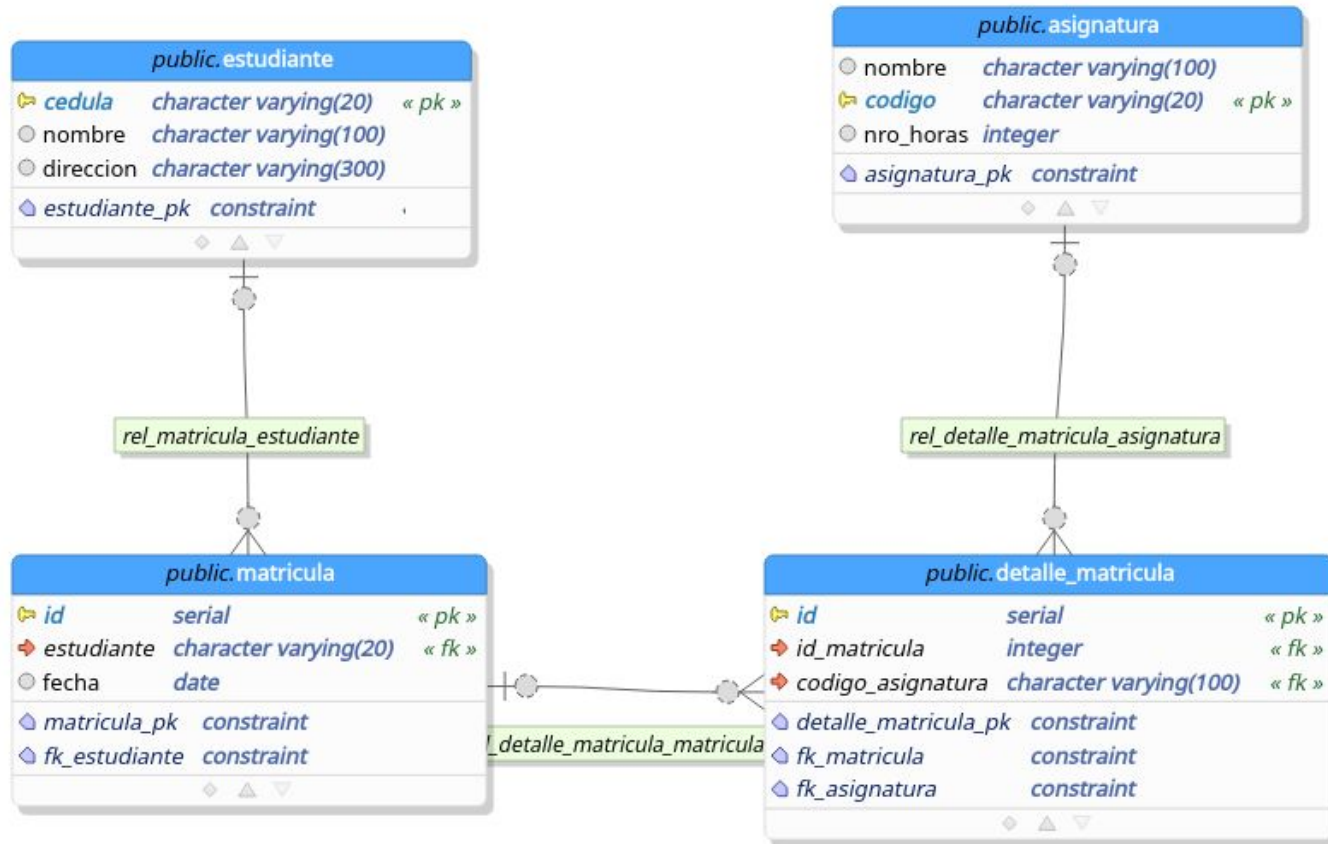
Ejecutamos el Script desde la base de datos  
ver video: Exportar un modelo en pgmodeler a una  
base de datos

[https://youtu.be/acdvB\\_ADW\\_s](https://youtu.be/acdvB_ADW_s)

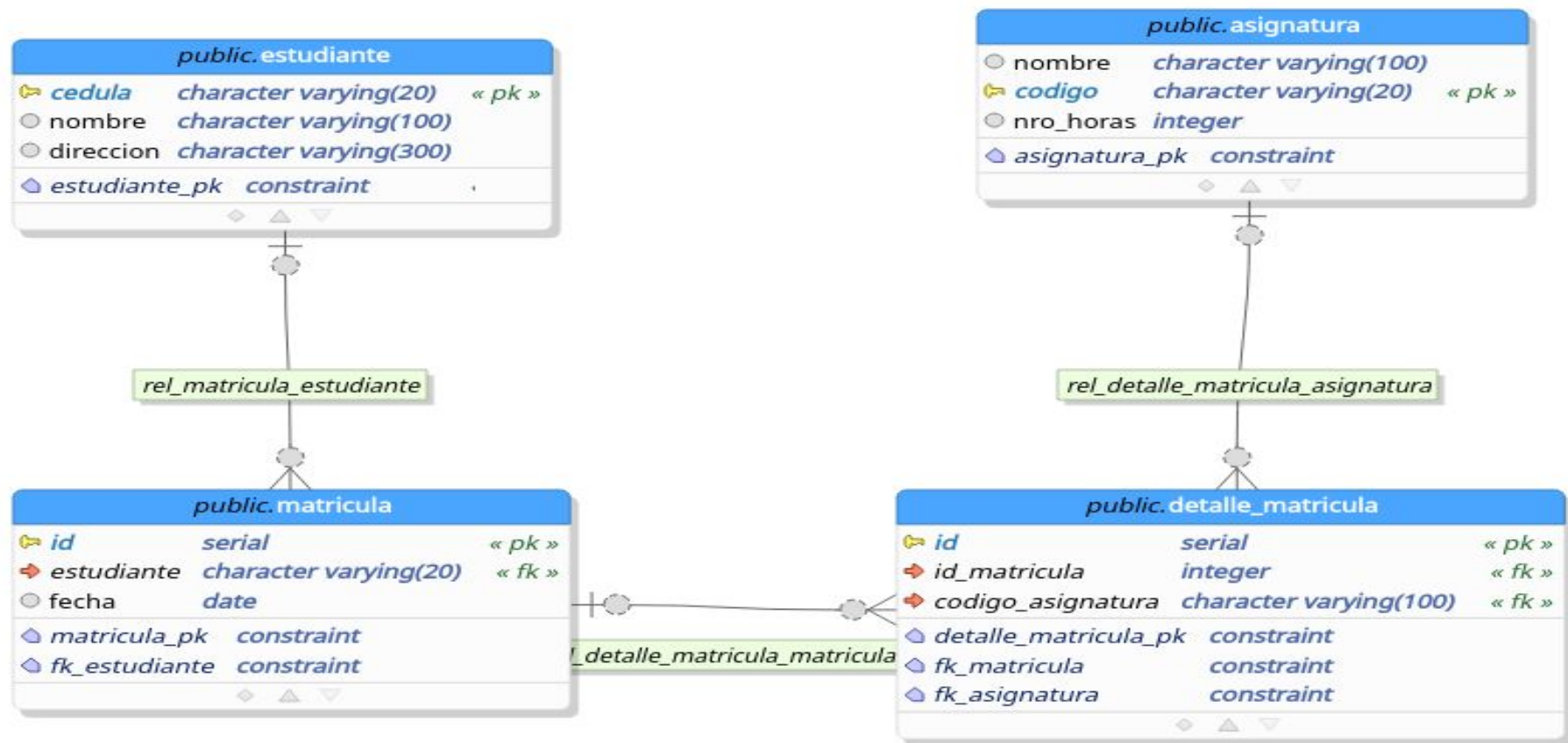
ver video: generar un modelo desde una base de  
datos

<https://youtu.be/zi5rpdOkGag>

# Realizar un Modelo con PGModeler



# Ejercicio de Matrícula en PGModeler





# Diccionario de Datos

**Objetivo: Dar al usuario de un mejor entendimiento sobre la base de datos y su estructura**

# Diccionario de Datos generado con pgmodeler

**Database:** matricula

## Tables

public.asignatura  
public.detalle\_matricula  
public.estudiante  
public.matricula

### public.asignatura

Table

*Registra todas las asignaturas que tienen las mallas curriculares*

Name	Data type	PK	FK	UQ	Not null	Default value	Description
nombre	character varying(100)						Nombre de la Asignatura
codigo	character varying(20)	✓			✓		puede ser alfanumerico
nro_horas	integer						registra las horas que se ven en el semestre de esa asignatura

### Constraints

Name	Type	Column(s)	References	Expression	Description
asignatura_pk	PRIMARY KEY	codigo			


























# Script generado con pgmodeler

```
-- object: public.cliente | type: TABLE --
-- DROP TABLE IF EXISTS public.cliente CASCADE;
CREATE TABLE public.cliente (
  * cedula character varying(20) NOT NULL,
  * nombre character varying(100),
  * estado integer,
  * CONSTRAINT cliente_pk PRIMARY KEY (cedula)
);
-- ddl-end --
COMMENT ON TABLE public.cliente IS E'Permite guardar la informacion de las personas que compran en la empresa';
-- ddl-end --
COMMENT ON COLUMN public.cliente.estado IS E'1: Activo, 2: inactivo';
-- ddl-end --
ALTER TABLE public.cliente OWNER TO postgres;
-- ddl-end --

-- object: public.direccion | type: TABLE --
-- DROP TABLE IF EXISTS public.direccion CASCADE;
CREATE TABLE public.direccion (
  * id serial NOT NULL,
  * calle_principal character varying(100),
  * calle_secundaria character varying(100),
  * sector character varying(100),
  * cliente character varying(20),
  * CONSTRAINT direccion_pk PRIMARY KEY (id)
);
-- ddl-end --
COMMENT ON TABLE public.direccion IS E'Un cliente puede tener varias direcciones, por tanto esta tabla esta relacionada con el cliente';
-- ddl-end --
COMMENT ON COLUMN public.direccion.sector IS E'Nombre del sector donde vive';
-- ddl-end --
ALTER TABLE public.direccion OWNER TO postgres;
-- ddl-end --

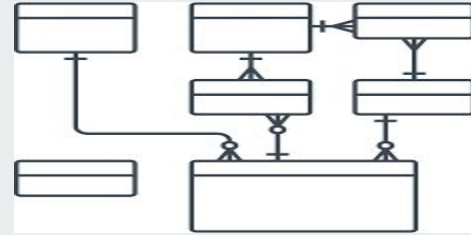
-- object: fk_cliente | type: CONSTRAINT --
-- ALTER TABLE public.direccion DROP CONSTRAINT IF EXISTS fk_cliente CASCADE;
ALTER TABLE public.direccion ADD CONSTRAINT fk_cliente FOREIGN KEY (cliente)
REFERENCES public.cliente (cedula) MATCH FULL
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --
```

# Generar una DB con pgmodeler

- ▼  matricula
  - >  Casts
  - >  Catalogs
  - >  Event Triggers
  - >  Extensions
  - >  Foreign Data Wrappers
  - >  Languages
  - ▼  Schemas (1)
    - ▼  public
      - >  Collations
      - >  Domains
      - >  FTS Configurations
      - >  FTS Dictionaries
      - >  FTS Parsers
      - >  FTS Templates
      - >  Foreign Tables
      - >  Functions
      - >  Materialized Views
      - >  Procedures
      - >  1.3 Sequences
      - ▼  Tables (4)
        - >  asignatura
        - >  detalle\_matricula
        - >  estudiante
        - >  matricula

# Clase Nro 2

## Bases de Datos avanzada



# Consideración al crear una Base de datos

**Corrección:** *Que cumpla con los requerimientos* ¿Hace lo que se le pide?

**Fiabilidad:** *Funcione bien* ¿Lo hace de forma fiable todo el tiempo?).

**Eficiencia:** *de respuesta Rápida,* (¿Qué recursos hardware y software necesito?).

**Integridad:** *correctitud y completitud de la información* (¿Se puede controlar su uso?).

**Facilidad de uso:** (¿Es fácil y cómodo de manejar?).

**Disponibilidad**

**Seguridad:**

**Auditable:**



Lo que el cliente dijo  
que necesitaba



Cómo el fabricante  
entendió lo que quería



Lo que el ingeniero de  
producto diseñó



Cómo entendió los  
requisitos el equipo



Lo que entendió el  
consultor de negocio



Cómo fue  
documentado



Lo que el cliente recibió  
en casa



Lo que se cobró al  
cliente



Cómo se diseñó el  
soporte y atención



Lo que el cliente  
realmente necesitaba

# Tipos de Integridad de datos

**La integridad de datos se refiere a la precisión, integralidad y coherencia general de los datos.** Cuando la integridad de los datos es segura, la información almacenada en una base de datos seguirá siendo completa, precisa y fiable por mucho tiempo que pase almacenada o por muchas veces que acceda uno a ella. La integridad de los datos también garantiza que sus datos estarán a salvo de fuerzas externas.

<https://www.talend.com/es/resources/what-is-data-integrity/>



# Tipos de Integridad de datos

Existen dos tipos de integridad de datos: la integridad física y la lógica

## Integridad física

La integridad física es la protección de la integridad y la precisión de los datos tal y como están almacenados y como son extraídos. Cuando tiene lugar una catástrofe natural, se produce un apagón o unos hackers alteran las funciones de una base de datos, se dice que se ha comprometido la integridad física. El error humano, la erosión del almacenamiento y toda una retahíla de incidencias también pueden imposibilitar que los gestores de tratamiento de datos, programadores de sistemas o aplicaciones y los auditores internos obtengan datos veraces.

## Integridad lógica

La integridad lógica conserva los datos sin ningún cambio, puesto que se emplean de forma distinta en una base de datos relacional. La integridad lógica protege a los datos del error humano y también de los hackers, pero de una forma muy distinta a la integridad física.

# Tipos de Integridad lógica

Existen cuatro tipos de integridad lógica.

## **Integridad de la entidad**

La integridad de la entidad se basa en la creación de unas claves primarias, o valores únicos, que identifican datos para asegurar que no aparezcan enumerados más de una vez y que no haya ningún campo de una tabla considerado nulo. Es una prestación de los sistemas relacionales que almacenan datos en tablas que pueden enlazarse y emplearse de formas muy distintas.

## **Integridad referencial**

La integridad referencial es una serie de procesos que aseguran que los datos se almacenen y se utilicen uniformemente. Las reglas integradas en la estructura de la base de datos sobre cómo se utilizan claves foráneas para garantizar que tan solo se produzcan cambios, incorporaciones o supresiones de datos adecuados. Las reglas pueden incluir restricciones que eliminen la entrada de datos duplicados, aseguren que los datos son veraces y/o impidan la entrada de datos no pertinentes.

# Tipos de Integridad lógica

## **Integridad de dominio**

La integridad de dominio es el conjunto de procesos que garantizan la veracidad de cada dato de un dominio. En este contexto, un dominio es un conjunto de valores aceptables que una columna puede contener. Puede incorporar restricciones y otras medidas que limiten el formato, tipo y cantidad de datos introducidos.

## **Integridad definida por el usuario**

La integridad definida por el usuario comprende las reglas y restricciones creadas por el usuario para adaptarse a sus necesidades particulares. En ocasiones con la integridad de entidad, referencial y de dominio no basta para salvaguardar los datos. A menudo, deben tenerse en cuenta determinadas reglas corporativas concretas e incorporarse en las medidas referentes a la integridad de los datos.

# Objetivos Generales de la Auditoría de BD

Es el proceso que permite medir, asegurar, demostrar, monitorear y registrar los accesos a la información almacenada en las bases de datos.

Disponer de mecanismos que permitan tener trazas de auditoría completas y automáticas relacionadas con el acceso a las bases de datos incluyendo la capacidad de generar alertas con el objetivo de:

- Mitigar los riesgos asociados con el manejo inadecuado de los datos.
- Apoyar el cumplimiento regulatorio.
- Satisfacer los requerimientos de los auditores.
- Evitar acciones criminales.
- Evitar multas por incumplimiento.

# Importancia de la Auditoría de BD

\_Toda la información financiera de la organización reside en bases de datos y deben existir controles relacionados con el acceso a las mismas.

- Se debe poder demostrar la integridad de la información almacenada en las bases de datos.
- Las organizaciones deben mitigar los riesgos asociados a la pérdida de datos y a la fuga de información.
- La información confidencial de los clientes, son responsabilidad de las organizaciones.
- Los datos convertidos en información a través de bases de datos y procesos de negocios representan el negocio.
- Las organizaciones deben tomar medidas mucho más allá de asegurar sus datos.

# Base de datos Auditables

Es el proceso que permite medir, asegurar, demostrar, monitorear y registrar los accesos a la información almacenada en las bases de datos incluyendo la capacidad de determinar:

- Quién accede a los datos.
- Cuándo se accedió a los datos.
- Desde qué tipo de dispositivo/aplicación.
- Desde qué ubicación en la Red.
- Cuál fue la sentencia SQL ejecutada.
- Cuál fue el efecto del acceso a la base de datos.

# Base de datos Seguras

La seguridad de datos es el conjunto de medidas adoptadas para evitar la corrupción de los datos. Incorpora el uso de sistemas, procesos y procedimientos que mantienen los datos inaccesibles para los demás, que podrían usarlos de forma perjudicial o distinta a la prevista. Las infracciones en materia de seguridad pueden ser pequeñas y fáciles de contener o grandes y provocar daños considerables.

# Modelo de una Base de datos Auditables y con control de usuarios y permisos

***Ver imagen***

<https://drive.google.com/file/d/1bOTkUxRzpr3VnQOE9Gnm9zu5Gl4GO3vS/view?usp=sharing>



# Grupos de proyecto

## Grupo 1:

*Mauricio Matango  
Kevin Guachagmira  
Henry Alvarado*

## Grupo 2:

*Francisco Jumbo  
Kevin Quemag  
Anthony Santillan*

## Grupo 3:

*Michael Pastrana  
Brayan Andrade  
Bryan Rivera*

## Grupo 4:

*Paola Acosta  
Erick Damian  
Edison Molina*

## Grupo 5:

*Steven Chinchin  
Joel Valencia  
Jennifer Guañuna*

## Grupo 6:

*Alberto Cadena(No tra  
Bryan Pérez  
Israel Salazar*

## Grupo 7:

*Maria jose  
Emerson Alexander Larco  
Martinez  
Jennifer Fernanda Defaz  
Guaman*

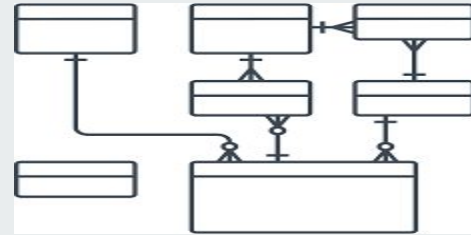
## Grupo 8: No matriculados

*Mario Alexander Soledispa  
Aguilar*

*Jhonathan Alejandro  
Chiliquing*

# Clase Nro 3

## Práctica Herramienta CASE pgmodeler





"Si un hombre tiene hambre no le des un pez, enséñale a pescar."

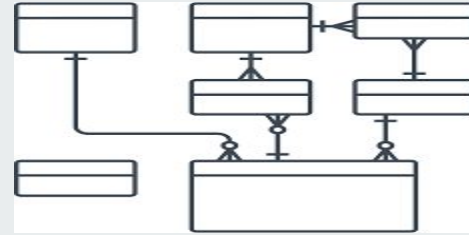
**Proverbio chino**

# Realizar el Diseño de la Base de datos

Se requiere automatizar el proceso de **atención al paciente para una cadena de hospitales**, por lo que se requiere que usted realice el diseño de la base de datos. Se conoce que se consta de **varios hospitales**, cada uno de los cuales integra uno o varios **servicios** (“traumatología”, “medicina interna”, “urología”, etc.), en los que son atendidos los **pacientes**. Puesto que no todos los hospitales disponen de los **mismos servicios**, en la ficha de cada hospital figura la **lista de servicios disponibles** y, para cada uno de ellos, **el número de camas que tiene** (si procede, pues **no** todos los servicios hospitalarios disponen de camas). Otros datos que figuran en la ficha de cada hospital son su código identificativo (codHospital), el nombre la ciudad en que está ubicado, el teléfono, y el nombre del director (**uno de los médicos adscritos a dicho hospital**). por cada medico se guardan datos personales (cedula, apellidos-nombre y fecha de nacimiento), el hospital al que está adscrito, y la lista de servicios hospitalarios en los que trabaja (puede desarrollar su actividad en varios servicios, del mismo o de diferentes hospitales). Todo ciudadano que ha utilizado alguna vez los servicios sanitarios tiene asociado un documento o “Historia Clínica”, identificado por un número único (codHist) dentro de la red sanitaria. En cada historia clínica figuran los datos personales del paciente (cedula, apellidos-nombre, fecha de nacimiento, número de Seguridad Social y otros datos opcionales), junto con la lista de todas las visitas médicas realizadas. En cada una de estas visitas consta la fecha y hora, el hospital y servicio en que ha sido atendido, el médico, y una breve descripción del diagnóstico y tratamiento realizados. Si el paciente es ingresado, se hace constar, además, el nº de habitación y la fecha en que abandona el hospital. **El interés de esta información no es sólo estadístico, sino que debe permitir conocer en todo momento el nº de camas libres de cada servicio de un hospital.** Para simplificar la identificación de los diferentes servicios, se ha decidido utilizar acrónimos de los nombres (idServicio), si bien también se desea tener almacenado su nombre completo, y un comentario opcional. Hay que tener en cuenta que un mismo servicio se puede ofrecer en hospitales distintos.

# Clase Nro 4

## Evaluación de Modelado Herramienta CASE pgmodeler



# Evaluación de Modelado

1. Control de Alquiler de Vehiculo: Considere que un cliente se le pueden alquilar varios vehículos en una misma fecha
2. Control de ventas de boletos de avión: un cliente se les puede comprar varios boletos, cada boleto indica los datos del pasajero, fecha de vuelo, destino,nro. asiento
3. Control de Préstamos de Libros: Considere que a un lector se le puede prestar varios libros en cada prestamo.
4. Control del citas de una mascota: Una mascota tiene un codigo de historia clínica , en cada cita se anexa los datos de la cita como fecha de consulta, diagnostico, tratamiento, veterinario que atendio, se desea conocer todo la historia de la mascota.
5. Control de visitas a un centro penitenciario: un privado de libertad puede recibir muchas visitas por dia, y muchas visitas durante toda el tiempo que este en prisión.
6. Alquiler de Bicicleta: Considere que un cliente se le pueden alquilar varios Bicicletas
7. Control de Pedidos: un cliente puede pedir muchos productos
8. Control de servicios(corte de cabello, aplicación de tinte, cepillado, etc.) que realiza un empleado en un salón de belleza
9. Control de todas las recetas de un cocinero, cada receta tiene muchos productos
10. Control de Compras: un cliente puede comprar muchos productos
11. Alquiler de película: un cliente puede alquilar muchas películas
12. 12. Control de ingreso de todos los empleados a una empresa, cada dia ingresan muchos empleados a la empresa.

# Evaluación de Modelado

13. Control de entrega de activos fijos a los empleados.
14. Control de transacciones que realiza un cliente con su cuenta,
15. Sistema que permita generar el un estado de cuenta (datos del cliente).
- 16 Sistema que permita mostrar todos los servicios (frenos, motor, cambio de aceite) que se le ha realizado a un vehículo.  
(un vehículo se le puede realizar muchos servicios en diferentes fechas)
17. Sistema que permita el control de distributivo docente: Considerar que los distributivos se realizan semestralmente y un distributivo contiene información de los docentes la asignatura que va a impartir .
18. Sistema de Ventas para un vivero: a un cliente se le pueden vender muchas plantas
19. Sistema de control de tareas de los estudiantes: se requiere conocer las tareas que el estudiante ha entregado el lunes pasado. (en un día el estudiante puede entregar muchas tareas.
20. Sistema de control de solicitudes realizada por un estudiantes: se requiere conocer todas que el estudiante ha realizado solicitudes, que tipo de solicitados han realizado y el estado de la solicitud.
21. Control del citas odontologicas: Cuando una persona acude por primera vez al centro se le asigna un codigo de historia clinica, el cual se va actualizando cada vez que el paciente acude a una cita. Interesa saber todas las citas odontologicas

## 1. Realizar el Diseño de la Base de Datos el tema indicado.

Debe estar Normalizado (No redundancia de datos) ( 20 pts)

Debe permitir la integridad de datos (claves primarias, o valores únicos, claves foráneas, precisión, integralidad y coherencia general de los datos) ( 10 pts)

Debe permitir guardar históricos de cambios ( 10 pts)

Debe contemplar la parte de seguridad. ( 10 pts)

Debe contemplar Herencia de tablas ( 10 pts)

Debe estar documentado cada tabla y cada campo ( 20 pts)

Exportar el diseño a la base de datos

Plantearse un caso e ingresar datos los datos en la base de datos ( 20 pts)

## 2. Entregables

-Entregar en un documento de Word las imágenes de cada tabla con el ingreso de Información,

**Nota** en caso de considerar nuevos requerimientos, escribirlos en el documento word.

- La imagen del Diseño de la base de datos

-Entregar el diccionario de datos



# Evaluación de Modelado



ANTHONY SEBASTIAN ...



HENRRY DAVID ALVARA...



BRAYAN ANIBAL ANDR...



ISRAEL SEBASTIAN SAL...



BRYAN ANDRES PEREZ ...



JENIFFER ARACELY GU...



BRYAN FERNANDO RIV...



JOEL ALFREDO VALEN...



DALTON FRANCISCO J...



KEVIN ALEXANDER GU...



EDISON MAURICIO MO...



KEVIN ANDRES QUEMA...



ERICK JOEL DAMIAN C...



LUIS MAURICIO MATAN...



TANIA PAOLA ACOSTA ...



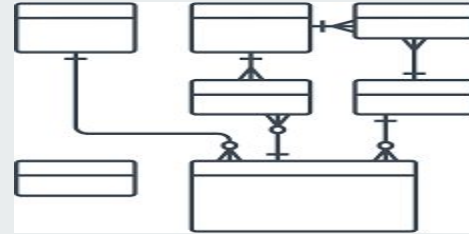
STEVEN ALEXANDER C...



MARIA JOSE ORTEGA P...

# Clase Nro 5

## VISTAS



**Al completar esta lección usted podrá entender los siguientes puntos:**

- ¿Qué es una Vista?
- ¿Cuál es su función?
- ¿Donde serán utilizadas?
- ¿Cómo recuperar datos?

## ¿Qué es una Vista?

**Es una estructura lógica que permite visualizar un grupo de datos que provienen de una o varias tablas u otras vistas.**

- No contiene datos propios, estos provienen de otras tablas con datos reales.
- No permite la inserción, actualización y eliminación de datos, solo la lectura.
- Su utilización es como el de una tabla, podemos usar cualquier sentencia de tipo `SELECT` sobre ellas.
- Prácticamente es una tabla virtual que proviene de la instrucción `SELECT`.

**Podemos utilizar estas estructuras para realizar las siguientes tareas:**

- Una o varias vistas pueden conformar una serie de pantallas del sistema de información.
- Reportes que no requieren un procesamiento complejo.

## ¿Que nos permite hacer?

- Restringen la información antes de ser mostrada al usuario final.
- Permite hacer consultas simples para aquellas que son complejas debido a que provienen de múltiples tablas o vistas gracias a la utilización de los JOIN.
- Permite unir datos divididos, desmoralizando un grupo de tablas.
- El cambio de la estructura física de la base de datos altera el resultado final de una consulta, esto se puede evitar con el uso de las vistas, el cual mantiene la misma estructura para otros sistemas a pesar que la consulta cambie internamente.

**Filtran el contenido a mostrar como un nivel mas de seguridad.**

- Permite oculta columnas y registros que no son deseados mediante condiciones.
- El uso de permisologías de usuario y de grupos permiten restringir la información a visualizar.

**De los puntos mencionados anteriormente, se puede apreciar lo siguiente:**

- Siempre se muestran los datos actualizados.
- Simplifica el uso de consultas complejas.
- Simplifica la representación de los datos ofreciendo mas sentido lógico.
- Define un nivel mas de seguridad.
- Aísla las aplicaciones de la Base de Datos.
- Permite mayor flexibilidad.



**De los puntos mencionados anteriormente, se puede apreciar lo siguiente:**

- No se pueden utilizar las sentencias **INSERT**, **UPDATE** y **DELETE** sobre la vista para alterar los datos.
- No mejora el rendimiento.

```
CREATE [ OR REPLACE ] VIEW name AS query
```

## Símbolo Significado

OR REPLACE

Permite remplazar la vista actual por una nueva, sin necesidad de eliminar y volverla a crear.

**La siguiente vista permite visualizar todas las películas que pertenecen al género de comedia.**

```
CREATE VIEW comedies AS SELECT *  
FROM films  
WHERE kind = 'Comedy';
```

**Puede remplazar una consulta contenida dentro de una vista de la siguiente forma:**

```
CREATE OR REPLACE VIEW comedies AS  
    SELECT code, title FROM films  
    WHERE kind = 'Comedy';
```

# Recuperar los datos

**Use la Vista de la misma forma que una tabla para recuperar los datos.**

```
SELECT * FROM comedies;
```

## Eliminar una Vista

**Utilice el siguiente comando para eliminar una Vista.**

```
DROP VIEW comedies;
```

```
create view v_credencias_usuarios as  
select usuario,clave from usuario
```

```
select * from v_credencias_usuarios
```

```
drop view v_credencias_usuarios
```

**Al completar esta lección usted podrá entender los siguientes puntos:**

- Conocer el propósito de las Vistas.
- Crear, modificar y eliminar Vistas.
- Implementar las vistas dentro de una consulta.



Al su tema de proyecto,

crear las 10 vistas que cumpla con el rol que usted tiene asignado (administrado, cajero o cliente)

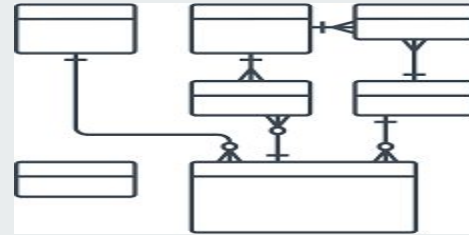
Trabajo individual, indicar el rol.

Entregar el modelo de la base de datos

Listado de requerimiento. Por cada requerimiento mostrar la vista creada, mostrar el resultado de la ejecución ) la base de datos debe tener datos

# Clase Nro 6

## Esquemas



1. Proponer los esquemas que van a utilizar en su proyecto
  - Definir el objetivo de cada esquema
  - mostrar captura de pantalla de los esquemas con sus respectivas tablas
2. Modificar los SQL realizados en las Vistas adaptados a estos nuevos cambio

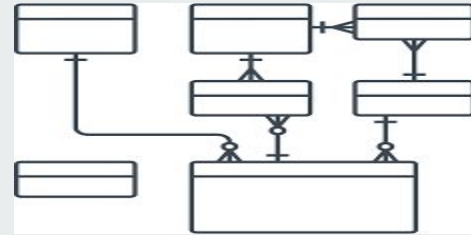
# Clase Nro 7

## Integridad de Datos

Integridad de la entidad

Integridad referencial

Integridad de dominio



# Integridad en bases de datos

- Al diseñar e implementar bases de datos, se debe prestar atención a la integridad de los datos y a cómo mantenerlos.

## ¿Qué es Integridad de Datos?

- La integridad de datos es un término usado para referirse a la exactitud y fiabilidad de los datos.

# Tipos de Integridad en bases de datos

1. Integridad de la entidad
2. Integridad referencial
3. Integridad de dominio
4. Integridad definida por el usuario

# 1. Integridad de la entidad:

## Tipos de restricciones

Las tablas pueden contener restricciones de integridad o reglas que limitan el tipo de dato que puede almacenarse en una tabla, fila o columna. Existen 5 tipos de restricciones de integridad, las cuales son:

RESTRICCIÓN **NOT NULL** (NO NULO)

RESTRICCIÓN **PRIMARY KEY** (CLAVE PRIMARIA)

RESTRICCIÓN **FOREIGN KEY** (CLAVE FORÁNEA)

RESTRICCIÓN **UNIQUE** (ÚNICA)

RESTRICCIÓN **CHECK** (CHEQUEO)



# 1. Integridad de la entidad:

## EJEMPLO

Se tiene la tabla estudiante en la que se debe contemplar los siguientes criterios de integridad

- El nombre del estudiante no puede ser NULL
- La cédula es el campo primario
- El género tiene una restricción con una clave foránea
- El número de teléfono es único
- El Promedio académico del estudiante no puede aceptar valores negativos.

Ejecutar el siguiente Código

# 1. Integridad de la entidad:

## La restricción NOT NULL (No nulo)



La restricción **NOT NULL** (No nulo), evita que un valor nulo sea ingresado en la columna. Por defecto, la clave primaria se define como NOT NULL. Todas las otras pueden ser NULL, a menos que la columna sea definida explícitamente como NOT NULL.

# 1. Integridad de la entidad:

## La restricción NOT NULL (No nulo) EJEMPLO

-El nombre del estudiante no puede ser NULL

```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK (promedio_academico > 0)
)
```



```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK
    (promedio_academico > 0)
)
```

# 1. Integridad de la entidad:

## La restricción PRIMARY KEY (CLAVE PRIMARIA)



Si una columna se define como clave primaria para una tabla, entonces dicha columna no puede tener un valor NULL (debe especificarse de manera explícita). este tipo de restricción asegura que no exista valores duplicados en esa columna. Si un usuario trata de agregar un registro, cuyo valor en la columna de la clave primaria es idéntico al de otro registro en la tabla, el registro no será insertado

# 1. Integridad de la entidad:

## La restricción PRIMARY KEY (CLAVE PRIMARIA) EJEMPLO



-La cédula es el campo primario

```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK (promedio_academico >0)
)
```

```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY
cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK
promedio_academico >0)
```

# 1. Integridad de la entidad:

## La restricción FOREIGN KEY (CLAVE FORÁNEA)



Si un campo primario en una tabla es usado **en otra tabla**, entonces esa columna en particular es llamada clave foránea

La tabla que contiene la columna de la clave primaria es llamada **tabla padre**. La tabla que contiene la clave foránea se llama **tabla hija**.

# 1. Integridad de la entidad:

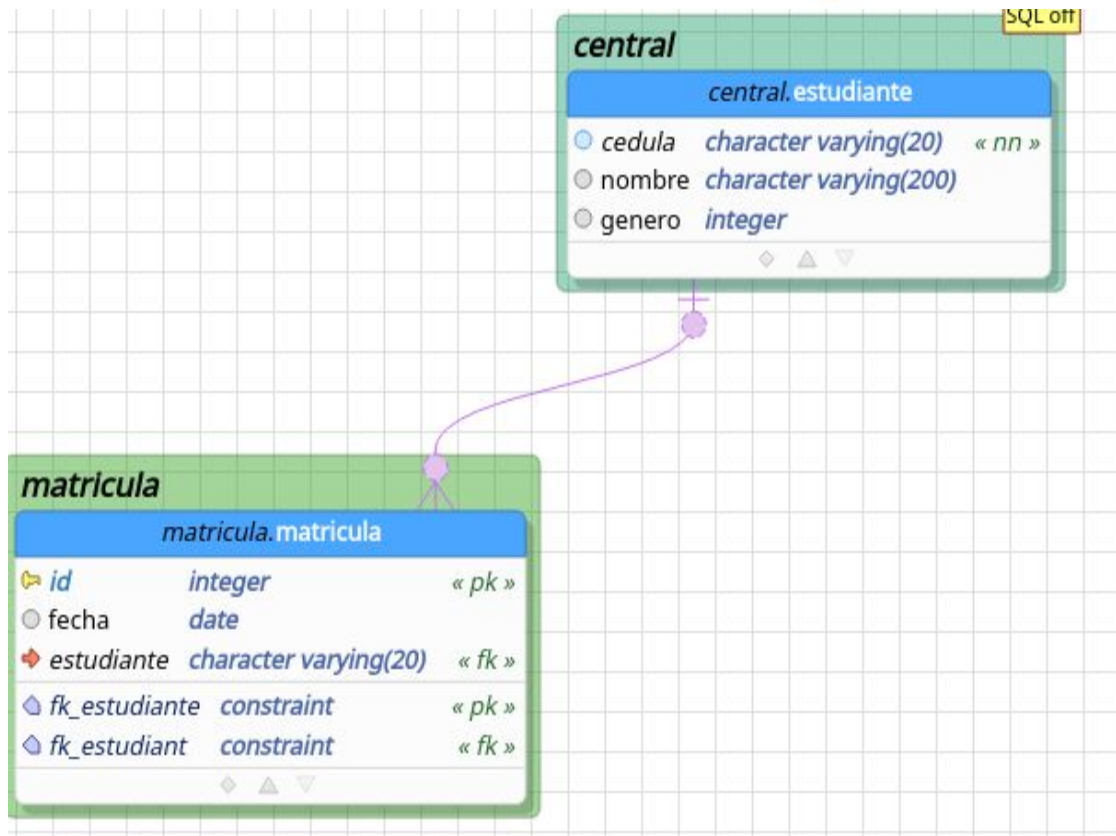


## La restricción FOREIGN KEY (CLAVE FORÁNEA)

### EJEMPLO

Cuál sería la tabla padre?

Cuál sería la tabla hija?



# 1. Integridad de la entidad:

## La restricción FOREIGN KEY (CLAVE FORÁNEA) EJEMPLO



El género tiene una restricción con una clave foránea

```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK (promedio_academico > 0)
)
```

```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK
    (promedio_academico > 0)
)
```



# 1. Integridad de la entidad:

## La restricción UNIQUE (ÚNICA)



Cuando se define esta restricción en la (s) columna(s) de una tabla, los datos no se pueden repetir.

# 1. Integridad de la entidad:

## La restricción UNIQUE (ÚNICA) EJEMPLO

El número de teléfono es único



```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK (promedio_academico > 0)
)
```

```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK
    (promedio_academico > 0)
)
```

# 1. Integridad de la entidad:

## La restricción CHECK (CHEQUEO)

Permite verificar la validez de los datos ingresados en una tabla contra un conjunto de restricciones.

### **Limitaciones**

- Una restricción Check puede hacer referencia a un conjunto específicos de datos constantes

# 1. Integridad de la entidad:

## La restricción CHECK (CHEQUEO) EJEMPLO

-El Promedio académico del estudiante no puede aceptar valores negativos.

```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK (promedio_academico > 0)
)
```



```
CREATE TABLE central.estudiante
( cedula character varying(20) NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  CONSTRAINT pk_estudiant PRIMARY KEY (cedula),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK
    (promedio_academico > 0)
)
```

# 1. Integridad de la entidad:

## La restricción UNIQUE (ÚNICA) EJEMPLO



-La fecha de nacimiento debe ser mayor a la fecha de hoy

```
CREATE TABLE central.estudiante|
(  ci central.cedula NOT NULL,
  nombre character varying(200) NOT NULL,
  fecha_nacimiento date,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  provincia_nacimiento central.provincias,
  CONSTRAINT pk_estudian PRIMARY KEY (ci),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK (promedio_academico >0),
  CONSTRAINT ch_fecha_nac CHECK (fecha_nacimiento>=current_date)
```

```
CREATE TABLE central.estudiante3
( ci central.cedula NOT NULL,
  nombre character varying(200) NOT NULL,
  fecha_nacimiento date,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  provincia_nacimiento central.provincias,
  CONSTRAINT pk_estudian PRIMARY KEY
  CONSTRAINT fk_genero FOREIGN KEY (g
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK
    (promedio_academico >0),
  CONSTRAINT ch_fecha_nac CHECK
    (fecha_nacimiento>=current_date)
```

# 1. Integridad de la entidad:

La restricción UNIQUE (ÚNICA)

## EJEMPLO

-La fecha de nacimiento debe ser mayor a la fecha c

```
INSERT INTO public.estudiante(  
cedula, nombre, fecha_nacimiento)  
VALUES ('9','Yogledis', '27-01-2021')
```

```
select * from estudiante  
alter table estudiante add constraint  
check_fecha_nacimiento check  
(current_date>=fecha_nacimiento)
```



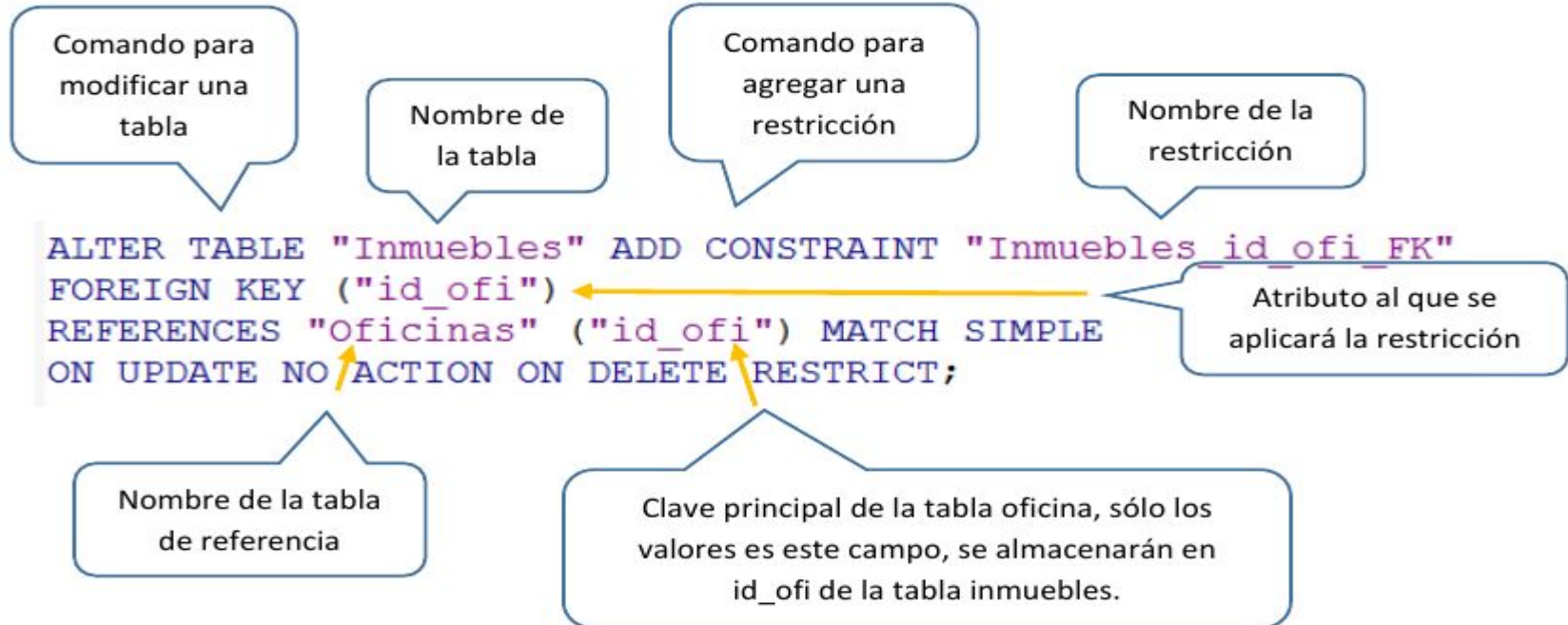
CREATE TABLE

```
( ci_central.cedula NOT NULL,  
.....  
nombre character varying(200) NOT NULL,  
fecha_nacimiento date,  
telefono character varying(20),  
promedio_academico numeric,  
genero integer,  
provincia_nacimiento central.provincias,  
CONSTRAINT pk_estudian PRIMARY KEY  
CONSTRAINT fk_genero FOREIGN KEY (g  
REFERENCES ignug.catalogo (id),  
CONSTRAINT uk_telefono unique (telefono),  
CONSTRAINT ch_promedio CHECK  
(promedio_academico >0),  
CONSTRAINT ch_fecha_nac CHECK  
(fecha_nacimiento>=current_date)  
)
```

## 2. Integridad referencial

se refiere a las relaciones.

- Para lograrlo se aplica **clave foráneas**



### 3. Integridad de dominio

- Los dominios son reglas que describen los valores de un campo. Proporcionan métodos para forzar la integridad de los datos

Los dominios de atributos se utilizan la limitar los valores permitidos en cualquier atributo concreto de una tabla.

Un dominio es una declaración de valores de atributos aceptables. cuando se asocia un dominio a un campo de atributo,es dedir no aceptará ningún valor que no éste en dicho dominio.

El uso de dominios ayuda a garantizar la integridad de los datos, al limitar las opciones de valores de un campo determinado.



### 3. Integridad de dominio:Ejemplo

-creamos un dominio para la cédula, que acepta números y guión

```
create domain central.provincias as character varying (20)
CONSTRAINT dom_provincia check (value in ('Pichincha','Cotopaxi','Tungurahua',
                                           'Carchi','Manabí','Guaya','Loja'))
```

```
create domain central.cedula as character varying (20)
```

1

```
create domain central.provincias as character varying (20)
CONSTRAINT dom_provincia check (value in
('Pichincha','Cotopaxi','Tungurahua','Carchi','Manabí','Guaya','Loja'))
```

```
create domain central.cedula as character varying (20)
```

### 3. Integridad de dominio:Ejemplo

-Usamos el dominio creado

```
CREATE TABLE central.estudiante4
( ci central.cedula NOT NULL,
  nombre character varying(200) NOT NULL,
  telefono character varying(20),
  promedio_academico numeric,
  genero integer,
  provincia_nacimiento central.provincias,
  CONSTRAINT pk_estudian PRIMARY KEY (ci),
  CONSTRAINT fk_genero FOREIGN KEY (genero)
    REFERENCES ignug.catalogo (id),
  CONSTRAINT uk_telefono unique (telefono),
  CONSTRAINT ch_promedio CHECK (promedio_academico >0)
)
```

### 3. Integridad de dominio:Ejemplo

-Probamos los dominios

```
INSERT INTO central.estudiante4(  
    ci, nombre,telefono, promedio_academico,genero,provincia_nacimiento )  
VALUES ('79','Yogledis','999',70.5,1,'Pichincha');
```

```
INSERT INTO central.estudiante4(  
    ci, nombre,telefono,  
    promedio_academico,genero,provi  
    ncia_nacimiento )  
VALUES  
('79','Yogledis','999',70.5,1,'Pichinc  
ha');
```

### 3. Integridad de dominio

Se refiere a la validez de las entradas para una columna determinada, puede incluir configuración de restricciones y reglas para definir el formato de datos o restringir el rango de valores posibles de entrada.

- Para lograrlo se aplica **restricciones CHECK, intervalo de valores, definiciones DEFAULT, NOT NULL**, entre otras

### 3. Integridad de dominio

La restricción **"check"** especifica los valores que acepta un campo, evitando que se ingresen valores inapropiados.

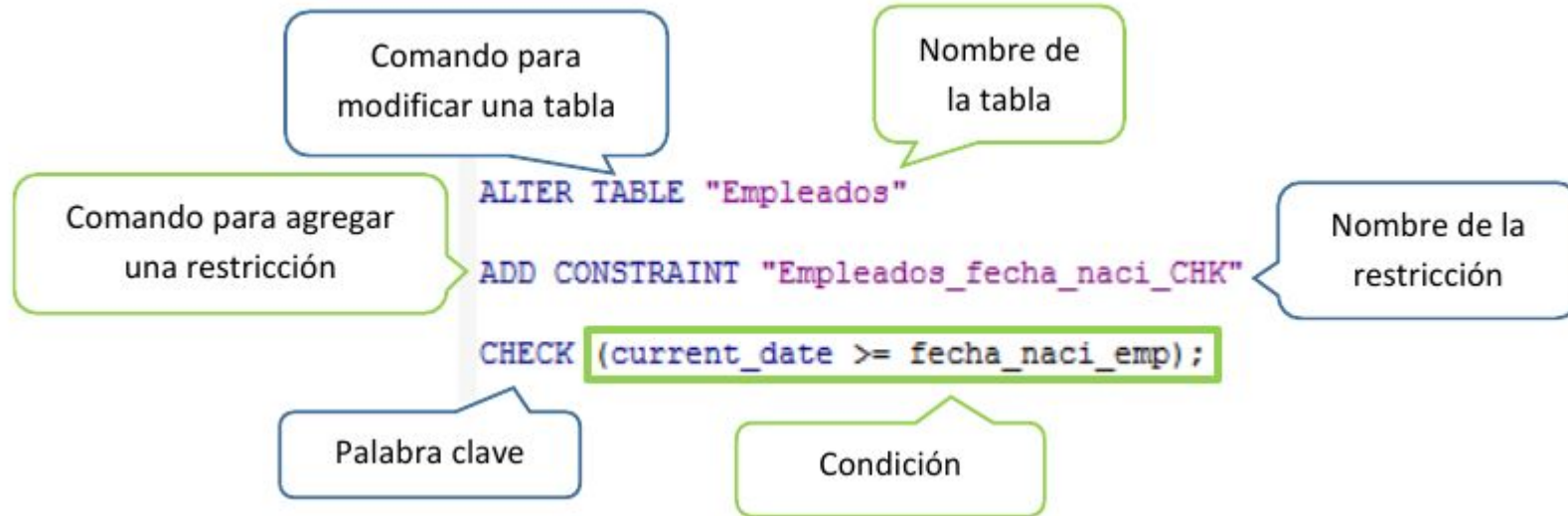
La sintaxis básica es la siguiente:

```
alter table NOMBRETABLA
```

```
add constraint NOMBRECONSTRAINT check CONDICION;
```

# Integridad de dominio: Ejemplo

## CHECK en ALTER TABLE



# Integridad de dominio: Ejemplo

CHECK en ALTER TABLE

```
INSERT INTO public.estudiante(  
cedula, nombre, fecha_nacimiento)  
VALUES ('9','Yogledis', '27-01-2021')
```

```
select * from estudiante  
alter table estudiante add constraint check_fecha_nacimiento  
check (current_date>=fecha_nacimiento)
```

# Integridad de dominio: Ejemplo

## 1. Ejemplo: edad > 0

```
create table prueba(id serial primary key, nombre varchar(20), edad integer)
```

```
drop table prueba
```

```
select * from prueba
```

```
alter table prueba add constraint limite_Edad check (edad>0)
```

```
insert into prueba values(1,'yogle',1)
```



# Integridad de dominio: Ejemplo

estudiante

estado\_civil

ci   nombre   fk\_edo

123   yogledis'   2   (S)

2   *Casado(a)*

234   Maria   3   (V)

1   Soltero

333   José   1   (C)

3   Viudo

# Integridad de dominio: Ejemplo

Ejemplo: CHECK en ALTER TABLE

Nombre que no sea vacío

```
insert into prueba (id,nombre,edad) values (2,"",3)
```

- alter table prueba add constraint check\_nombre\_obligatorio  
check (nombre>"")
-

## Tip Integridad de dominio: Ejemplo

### Ejemplo de Integridad de dominio

Otra Forma de aplicar CHECK en ALTER TABLE

```
ALTER TABLE prueba ADD CHECK (nombre>" )
```

# Integridad de dominio: Ejemplo

## Ejemplo de Integridad de dominio

**CHECK** (con múltiples columnas) en ALTER TABLE

```
select * from prueba
```

```
create table prueba(id serial primary key, nombre varchar(20),edad  
integer)
```

```
insert into prueba (id,nombre,edad) values (2,"",-3)
```

```
select * from prueba
```

```
delete from prueba
```

```
ALTER TABLE prueba ADD CHECK (nombre>" and edad>35)
```

# Integridad de dominio: Ejemplo

Ejemplo: Uso del CHECK

## Ejemplo sin Check

```
CREATE TABLE departamentos(id serial primary key, nombre  
varchar(255))
```

```
insert into departamentos values (-2,'Sistemas')
```

```
select * from departamentos
```

## Ejemplo con Check

```
drop table departamentos
```

```
CREATE TABLE departamentos(id serial NOT NULL, nombre  
varchar(255), CHECK (id>0) )
```

```
insert into departamentos values (2,'Sistemas');
```

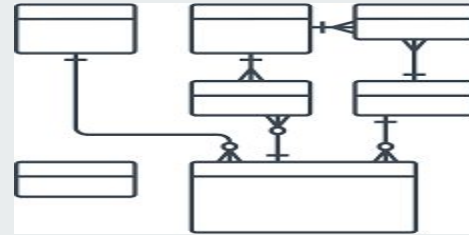
```
insert into departamentos values (-2,'Sistemas');
```

```
select * from departamentos;
```

# Clase Nro 8

## Integridad de Datos

Definida por el usuario  
Transacciones



## 4. Integridad definida por el usuario

- La integridad definida por el usuario le permite al usuario aplicar reglas comerciales a la base de datos que no están cubiertas por ninguno de los otros tres tipos de integridad de datos.

# Integridad definida por el usuario

- Transacciones
- Funciones /Procedimientos Almacenados



# Transacciones

Es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

# Transacciones

## Ejemplo de una transacción exitosa

Cuenta 1          cuenta 2

Nro\_cta:1111

Nro\_cta:222

cedula:12345

cedula:9999

saldo=1000

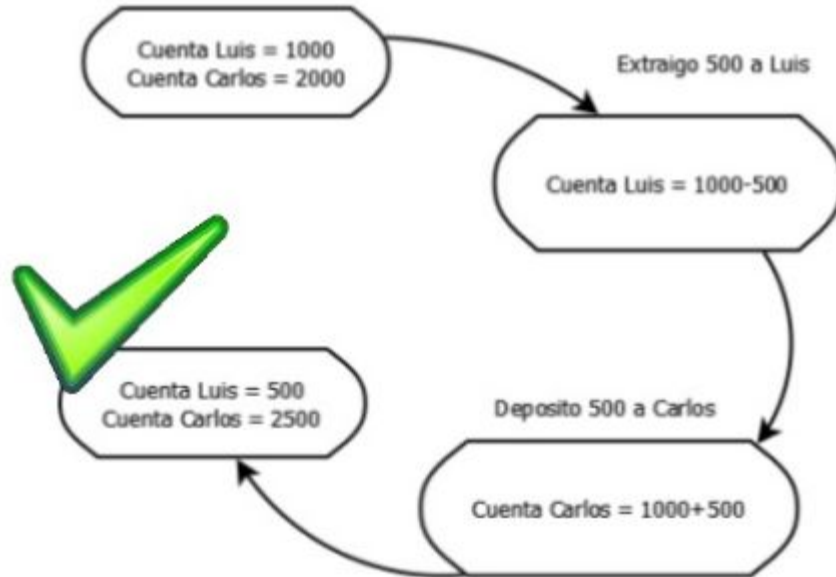
saldo=2000

debita\_cuenta1: update table cuenta set saldo=saldo-500  
where nro\_cuenta=1111

acredita\_cuenta2: update table cuenta set saldo=saldo+500  
where nro\_cuenta=2222

# Transacciones

Ejemplo de sin transacción con resultado exitoso



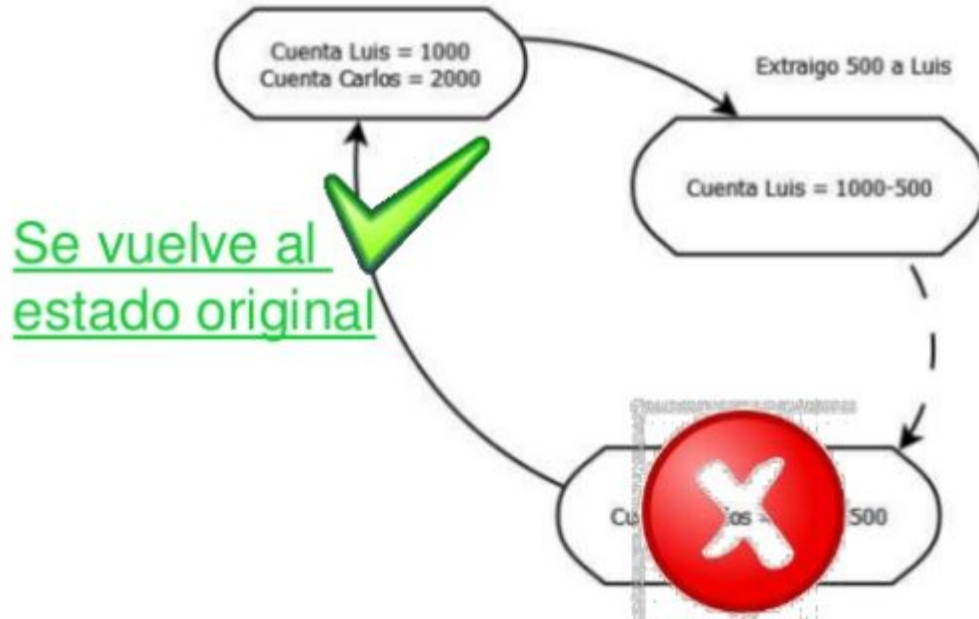
# Transacciones

Ejemplo de sin transacción con resultado inesperado



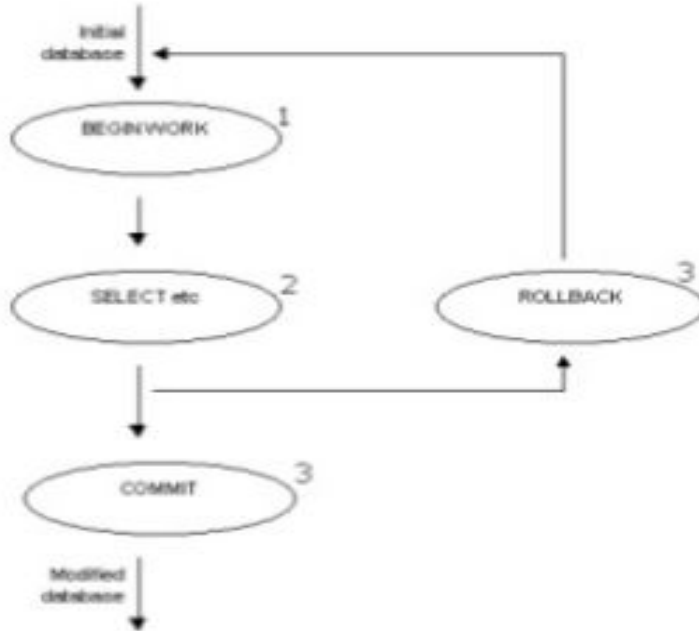
# Transacciones

Con transacción Error inesperado



# Transacciones

## Comandos COMMIT ROLLBACK



# Transacciones: Ejemplos

## EJEMPLOS

```
insert into cuentas values('1111','Luis',1000);
```

```
insert into cuentas values('222','Carlos',2000)
```

```
truncate cuentas
```

```
begin
```

```
    update  cuentas set saldo= saldo - 500 where nro_cuenta = '1111'
```

```
    update  cuenta set saldo= saldo + 500 where nro_cuenta = '222'
```

```
commit
```

```
rollback
```

# Clase 9

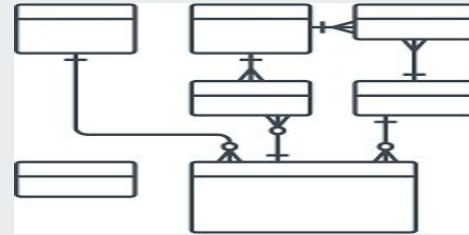
## Integridad de Datos

Definidas por el usuario

Funciones

/Procedimientos

Almacenados





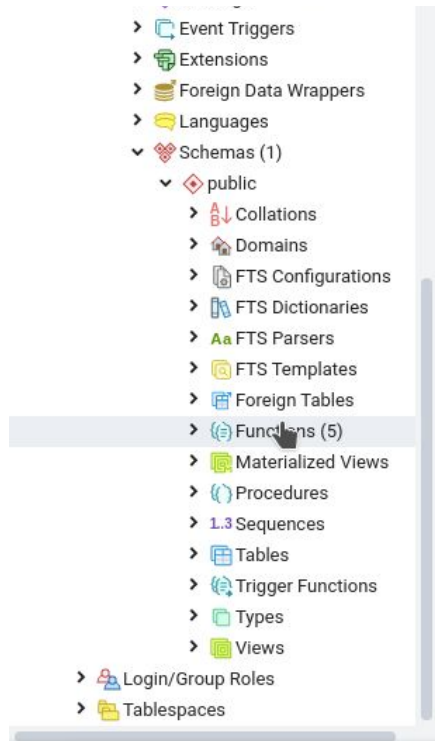
# Procedimientos almacenados (stored procedures) en PostgreSQL.

Un procedimiento almacenado se puede definir como un **programa, procedimiento ó función**, el cual está almacenado en la base de datos y listo para ser usado.

Documentación Oficial de Postgresql:

Capítulo 41. **Procedural Languages**

<https://www.postgresql.org/docs/current/xplang.html>



# Procedimientos almacenados (stored procedures) en PostgreSQL.

PostgreSQL permite que las funciones definidas por el usuario se escriban en otros lenguajes además de SQL y C.

Un procedimiento almacenado en PostgreSQL se puede escribir en múltiples lenguajes de programación. En una instalación por defecto de PostgreSQL podremos tener disponibles los siguientes lenguajes: PL/pgSQL, PL/Perl, PL/Tcl y PL/Python.

El único lenguaje que está disponible automáticamente es PL/pgSQL. Para utilizar PL/Perl, PL/Tcl o PL/Python tendremos que haber configurado/compilado PostgreSQL con estos parámetros `--with-perl --with-tcl --with-python`.

También existen muchos otros lenguajes disponibles como módulos adicionales, entre ellos, PL/Java, PL/PHP, PL/R, PL/Ruby, PL/Scheme y PL/sh, pero estos tienen que descargarse e instalarse por separado

# sintaxis

CREATE [OR REPLACE] FUNCIÓN nombre (argumentos)

RETURNS valor

AS \$\$

DECLARE

Variable ALIAS FOR \$número del argumento;

BEGIN

T

RETURN valor;

END;

\$\$Language ' plpgsql ';

# Funciones: cómo consultar y eliminar una función

- **CONSULTAR UNA FUNCIÓN**

**SINTAXIS SELECT NOMBREFUNCION()**

**SELECT myFuncion2();**

- **SINTAXIS PARA ELIMINAR UNA FUNCIÓN**

- **DROP FUNCTION myFuncion();**

- 

- **DROP FUNCTION myFuncion(int)**

-

# Tipos de Funciones

## **FUNCIONES SIN PARÁMETROS Y CON PARÁMETROS**

# Funciones con parámetro

- **Comando CREATE FUNCTION**

```
CREATE FUNCTION myFuncion(variable int) RETURNS int AS
$$
BEGIN
RETURN variable + 1;
END;
$$ LANGUAGE plpgsql;

SELECT myFuncion(5);
```

# Funciones sin parámetro

- **Comando CREATE FUNCTION**

```
CREATE FUNCTION myFuncion2() RETURNS int AS $$  
BEGIN  
RETURN 1 + 1;  
END;  
$$ LANGUAGE plpgsql;
```

```
SELECT myFuncion2();
```

# Funciones: Ejemplo crear una función que devuelva el cliente de una cuenta

Nombre de la  
función

Argumentos

```
CREATE FUNCTION public.f_prueba2(IN v_nro_cuenta character varying)
```

```
RETURNS character varying
```

```
LANGUAGE 'sql'
```

Especificación  
del  
lenguaje

Consulta  
SQL

```
AS $BODY$
```

```
select cliente from cuenta where nro_cuenta=v_nro_cuenta
```


```
$BODY$;
```



# Funciones: cómo consultar y eliminar una función

```
select f_prueba2('1111') ;
```

Data Output Explain

f_prueba2	
character varying	
12345	

```
drop function f_prueba2 ;
```

# Código del Ejemplo

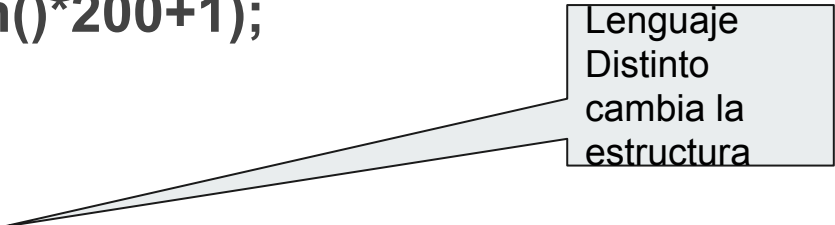
```
CREATE FUNCTION public.f_prueba2(IN  
v_nro_cuenta character varying)  
RETURNS character varying  
LANGUAGE 'sql'
```

```
AS $BODY$  
select cliente from cuenta where  
nro_cuenta=v_nro_cuenta  
$BODY$;
```

```
select f_prueba2('1111')  
drop function f_prueba2
```

# Funcione que permita retornar un numero aleatorio entre 0 y 200

```
CREATE or REPLACE FUNCTION myFuncion3() RETURNS int AS  
  
$$  
  
BEGIN  
  
RETURN trunc(random()*200+1);  
  
END;  
  
$$ LANGUAGE plpgsql;
```



Lenguaje  
Distinto  
cambia la  
estructura

# Función que devuelve un tipo de dato tabla

```
12 CREATE FUNCTION public.f_tabla(IN v_nro_cuenta character varying)
13     RETURNS setof cuenta
14     LANGUAGE 'sql'
15
16 AS $BODY$
17 select nro_cuenta,cliente,saldo from cuenta where nro_cuenta=v_nro_cuenta
18 $BODY$;
19
20 select f_tabla('1111')
21
```

Data Output Explain Messages Notifications

	f_tabla	
	cuenta	
1	(1111,12345,1000)	

```
CREATE FUNCTION public.f_tabla(IN v_nro_cuenta character varying)
  RETURNS setof cuenta
  LANGUAGE 'sql'

AS $BODY$
select nro_cuenta,cliente,saldo from cuenta where nro_cuenta=v_nro_cuenta
$BODY$;

select f_tabla('1111')
```

# Funcione que permita retornar la suma de dos números

```
CREATE or REPLACE FUNCTION suma2(n1 int,n2 int) RETURNS int AS
```

```
$$
```

```
declare
```

```
suma int;
```

```
BEGIN
```

```
suma=n1+n2;
```

```
RETURN suma;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

# Ejercicio: crear una funcion que retorne un número aleatorio entre un rango de datos

Creemos una Función llamado random between, esta función va a retornar un numero entero aleatorio, entre en rango definido en los parámetros, la función tiene 2 parámetro, donde indica el el rango mínimo y el rango máximo.

```
CREATE OR REPLACE FUNCTION random_between(low INT ,high  
INT)  
    RETURNS INT AS  
$$  
BEGIN  
    RETURN floor(random()* (high-low + 1) + low);  
END;  
$$ language 'plpgsql';
```

# Ejercicio: Crear Función que permita crear String aleatoriamente para Simular los

```
CREATE FUNCTION get_random_string(  
    IN string_length INTEGER,  
    IN possible_chars TEXT DEFAULT  
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'  
) RETURNS text  
LANGUAGE plpgsql  
AS $$  
DECLARE  
    output TEXT = '';  
    i INT4;  
    pos INT4;  
BEGIN  
    FOR i IN 1..string_length LOOP  
        pos := 1 + CAST( random() * ( LENGTH(possible_chars) - 1) AS INT4 );  
        output := output || substr(possible_chars, pos, 1);  
    END LOOP;  
    RETURN output;  
END;  
$$
```



```
CREATE OR REPLACE FUNCTION public.f_probar(IN v_cedula_usuario character varying)
RETURNS TABLE (
    cedula character varying,
    clave character varying
)
AS $$
BEGIN
    return query select per.cedula cedula , usu.clave clave from "Usuario"."Usuario" as usu
    join "Personal"."Persona" as per on usu.cedula_persona = per.cedula
    where per.cedula = v_cedula_usuario;

END;
$$ language 'plpgsql';

drop function f_probar

select f_probar('9356789787')
```

# Funcione que permite guardar en una tabla

```
CREATE OR REPLACE FUNCTION sp_cuenta_crear(IN p_nro_cuenta character varying, IN
p_cliente character varying, IN p_saldo numeric)
  RETURNS character varying AS
$BODY$
DECLARE
v_mensaje character varying;
BEGIN
    insert into cuenta(nro_cuenta,cliente,saldo)
        VALUES (p_nro_cuenta,p_cliente,p_saldo);
    v_mensaje='Cuenta registrada exitosamente';
    return  v_mensaje;
END;
$BODY$
LANGUAGE plpgsql

select sp_cuenta_crear('56789','77777',500);
```

sp_cuenta_crear
character varying

Cuenta registrada exitosamente
--------------------------------

## Ejercicio: Crear Función que permita crear String aleatoriamente para Simular los Nombre

```
Select get_random_string(  
    10,'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  
)
```




## Ejercicio: crear la tabla Estudiante

```
CREATE TABLE public."Estudiante"  
(  
  cedula character(20) COLLATE pg_catalog."default" NOT NULL,  
  nombre character(50) COLLATE pg_catalog."default",  
  edad integer, PRIMARY KEY (cedula)  
)
```

## Ejercicio: ingresar masivamente en la tabla estudiante

```
insert into "Estudiante"  
select  
"|| random_between(5000000,20000000)||'-'||  
random_between(0,9),get_random_string(  
    10,'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  
),random_between(17,50)  
from generate_series(0,100)
```

## Ejercicio: Utilicemos las funciones para crear algo con esto

	<b>cedula</b> [PK] character (20) 	<b>nombre</b> character (50) 	<b>edad</b> integer 
1	15525212-9	IJNMXMJKSB ...	43
2	16126931-2	AXCIIHOTUW ...	18
3	17603860-5	KJTIIETIKZT ...	44

# Tarea evaluada sobre funciones

## Actividad en Clases

1. Crear un procedimiento almacenado que permita guardar, modificar o eliminar, retorne un numero entero que muestre el resultado de la operación.

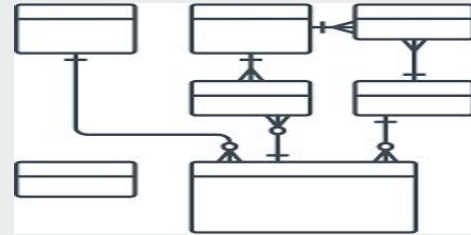
## Actividad trabajo autónomo

2. Desarrollar una interfaz gráfica que permita realizar las operaciones del CRUD a la tabla Cuenta, usando un procedimientos almacenados

# Clase Nro 10

## Revisión de proyecto: Integridad de Datos

Definida por el usuario  
Transacciones





# Tarea

Desarrollar una interfaz Cuentas que permita realizar un transferencia. Aplicar los conceptos de Transacciones

cuenta a Debitar

cuenta a Creditar

Saldo a Transferir

Tranferir Sin error

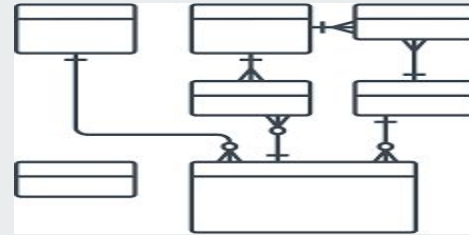
Tranferir con error

# Tarea

- Realizar una transferencia de 10 Dólares
- Mostrar los SQL Commit y Rollbak

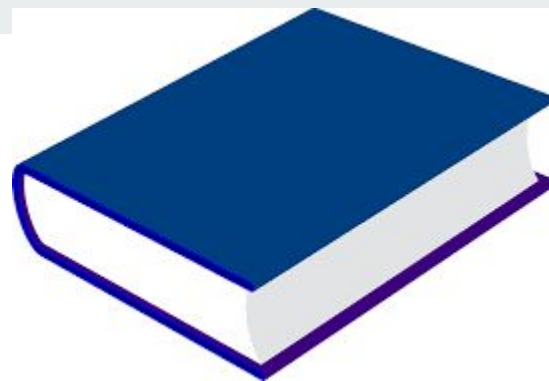
# Clase 11

## Índices



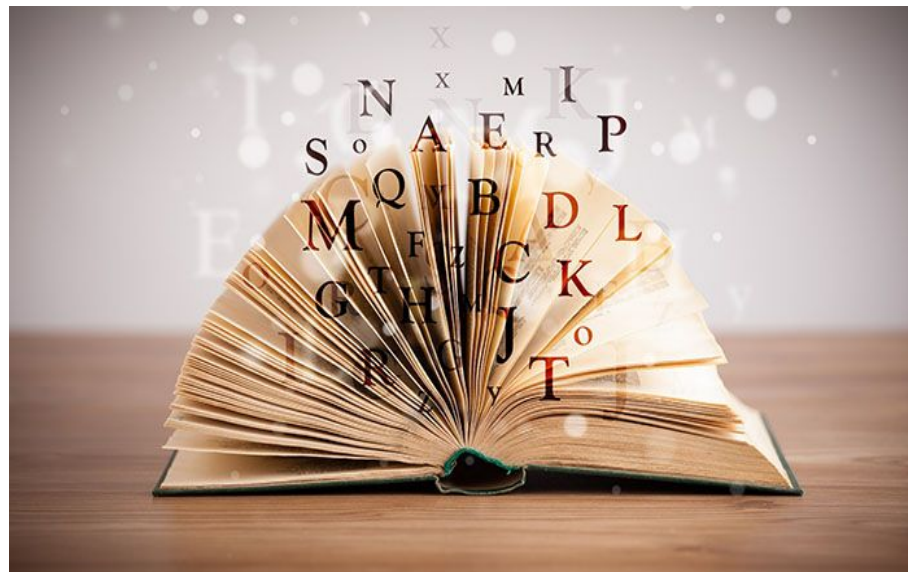
- Introducción
- Búsquedas del optimizador de Postgresql
- Pasos para optimizar una consulta SQL
- Ventajas de los índices
- Desventajas de los índices

# Introducción



## Índice

UNIDAD 1. Los Números Hasta el 10	1
UNIDAD 2. Contamos de 10 al 20	17
UNIDAD 3. Contamos de 20 al 30	29
UNIDAD 4. Contamos de 30 al 50	41
UNIDAD 5. Contamos de 50 al 100	61
UNIDAD 6. Nos Vamos de Compras	87
UNIDAD 7. Aprendemos la Hora	103
UNIDAD 8. ¿Dónde Estamos?	121



# Concepto de Índice en base de datos

1. Los índices nos permite mejorar el rendimiento de las bases de datos
2. Los índices permite que el servidor de base de datos encuentre y recupere filas específicas mucho más rápido de lo que podría hacerlo sin un índice
3. Los índices generan gastos a la base de datos en su conjunto, por lo que debe usarse con conciencia.

# Búsquedas del optimizador de PostgreSQL

SEQUENTIAL SCAN (BASICS)  
INDEX SCAN  
BITMAP SCAN

Fuente:

<https://www.cybertec-postgresql.com/en/postgresql-indexing-index-scan-vs-bitmap-scan-vs-sequential-scan-basics/>

# Búsquedas del optimizador de PostgreSQL

El Planificador/**Optimizador** es el módulo de **PostgreSQL** encargado de generar los posibles planes con los que se puede ejecutar una consulta dada y de elegir el menos costoso o relativamente eficiente.

**Con la instrucción Explain nos muestra el plan de ejecución de postgresql**

**EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON)**

```
select p.cedula,p.nombres,ca.nombre,eb.nombre
from persona p
join cuenta c
on p.cedula=c.cliente
join catalogo ca
on ca.id_catalogo=c.tipo_cuenta
join entidad_bancaria eb
on eb.ruc=c.ruc_entidad_bancaria
where p.nombres='yoli'
```



# Pasos para optimizar SQL

## 1. Se prueba la consulta sin índice con el explain

```
28 EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON)
29 select p.cedula,p.nombres,ca.nombre,eb.nombre
30 from persona p
31 join cuenta c
32 on p.cedula=c.cliente
33 join catalogo ca
34 on ca.id_catalogo=c.tipo_cuenta
35 join entidad_bancaria eb
36 on eb.ruc=c.ruc_entidad_bancaria
37 where p.nombres='yoli'
```


Data Output Explain Messages Notifications

QUERY PLAN	
1	[

```
EXPLAIN (ANALYZE, COSTS,
VERBOSE, BUFFERS, FORMAT
JSON)
select
p.cedula,p.nombres,ca.nombre,eb.
nombre
from persona p
join cuenta c
on p.cedula=c.cliente
join catalogo ca
on ca.id_catalogo=c.tipo_cuenta
join entidad_bancaria eb
on eb.ruc=c.ruc_entidad_bancaria
where p.nombres='yoli'
```

# Pasos para optimizar SQL

2. Se copia el json al <https://explain.dalibo.com/plan#> y se observa el rendimiento y tipos de búsquedas (tiempo y costo)

 **explain.dalibo.com**

Title

Plan (text or JSON)

```
"Shared Written Blocks": 0,  
"Local Hit Blocks": 0,  
"Local Read Blocks": 0,  
"Local Dirtied Blocks": 0,  
"Local Written Blocks": 0,  
"Temp Read Blocks": 0,  
"Temp Written Blocks": 0  
,
```






Query (optional)

Paste corresponding SQL query or drop a file

Submit

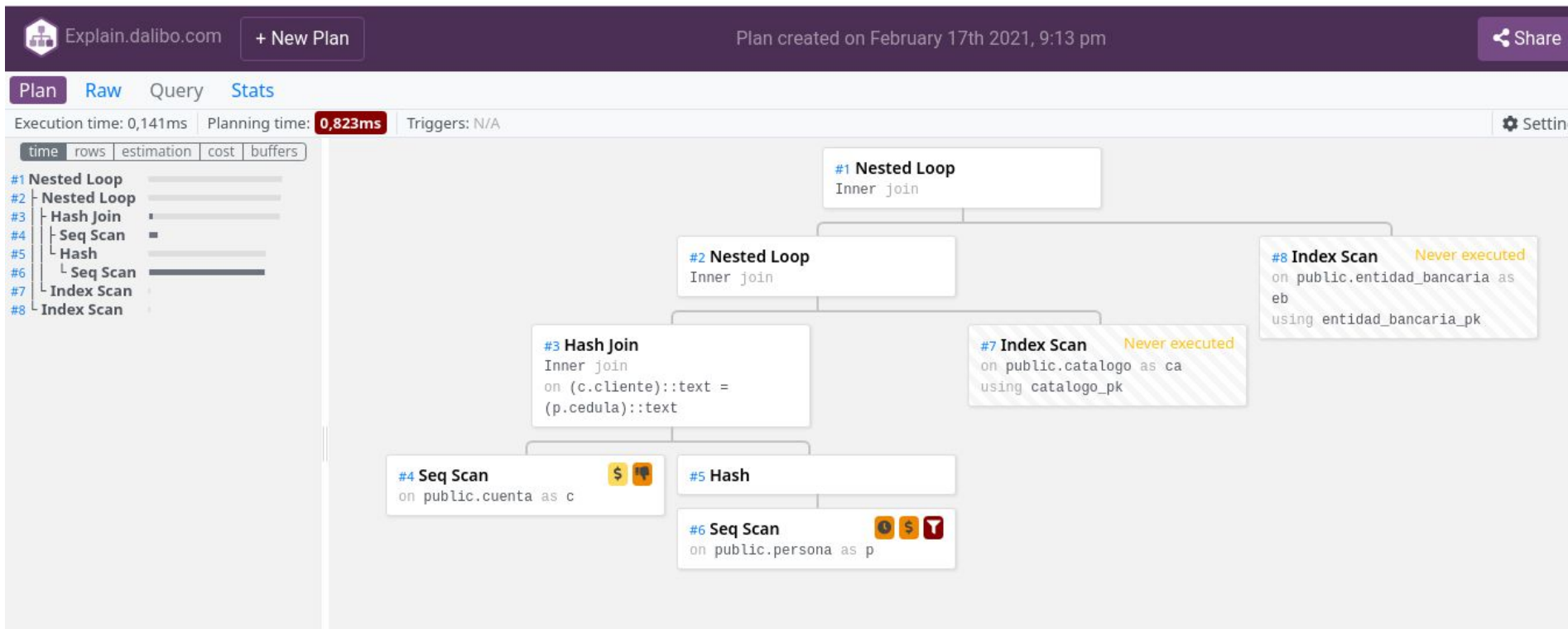
Sample Plans ▾

### Plans

Plan created on February 17th 2021, 9:13 pm created 2 minutes ago	
Plan created on February 17th 2021, 8:23 pm created about 1 hour ago	
Plan created on February 17th 2021, 8:19 pm created about 1 hour ago	
Plan created on February 17th 2021, 8:15 pm created about 1 hour ago	
Plan created on February 17th 2021, 8:04 pm created about 1 hour ago	

# Pasos para optimizar SQL

2. Se copia el json al <https://explain.dalibo.com/plan#> y se observa el rendimiento y tipos de búsquedas (tiempo y costo)



# Pasos para optimizar SQL

3. Se crea el índice

```
create index i_persona_nombre on persona(nombres)
```

```
create index i_persona_nombre on persona(nombres)
```

# Pasos para optimizar SQL

4. Mantenimiento a la tabla: Permite realizar un mantenimiento completo a la tabla y actualización de estadísticas de la tabla (Tamaño de campos, cant de registros, que índices hay) útil para que el planificador de postgres pueda tomar la mejor decisión

```
vacuum full analyze persona;
```

```
vacuum full analyze persona;
```

# Pasos para optimizar SQL

## 5. Medir el rendimiento de la consulta: repitiendo el paso 1 y 2

```
28 EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON)
29 select p.cedula,p.nombres,ca.nombre,eb.nombre
30 from persona p
31 join cuenta c
32 on p.cedula=c.cliente
33 join catalogo ca
34 on ca.id_catalogo=c.tipo_cuenta
35 join entidad_bancaria eb
36 on eb.ruc=c.ruc_entidad_bancaria
37 where p.nombres='yoli'
```

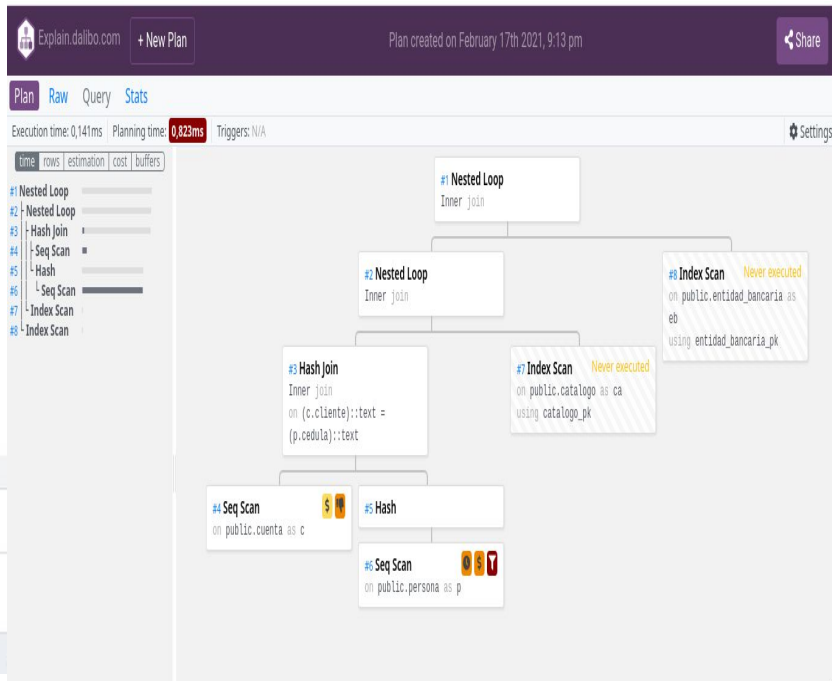
Data Output Explain Messages Notifications

QUERY PLAN

json

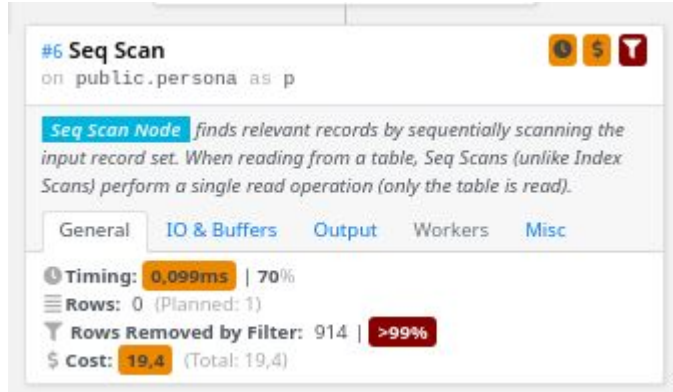
1

[



# Pasos para optimizar SQL

## 5. Generar conclusiones

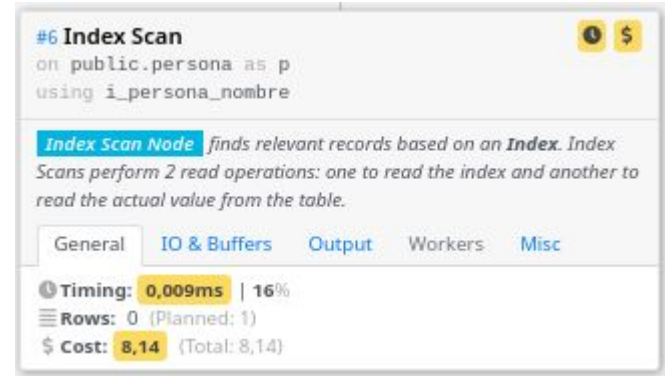


#6 Seq Scan  
on public.persona as p

**Seq Scan Node** finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General IO & Buffers Output Workers Misc

Timing: 0,099ms | 70%  
Rows: 0 (Planned: 1)  
Rows Removed by Filter: 914 | >99%  
Cost: 19,4 (Total: 19,4)



#6 Index Scan  
on public.persona as p  
using i\_persona\_nombre

**Index Scan Node** finds relevant records based on an **Index**. Index Scans perform 2 read operations: one to read the index and another to read the actual value from the table.

General IO & Buffers Output Workers Misc

Timing: 0,009ms | 16%  
Rows: 0 (Planned: 1)  
Cost: 8,14 (Total: 8,14)

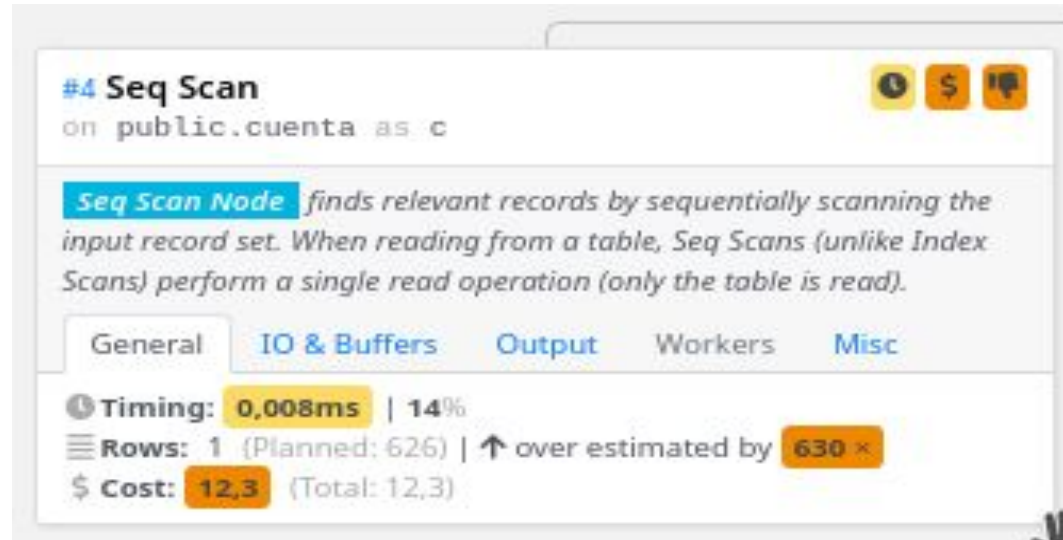
Se observa que con indices disminuye los costo de la consulta SQL, .....

sin embargo se observa que ahora hay una consulta que se puede optimizar.

# Pasos para optimizar SQL

## 5. Generar conclusiones

Sin embargo se observa que ahora hay una consulta que se puede optimizar.



Por tanto creemos los índices de la tabla cuenta



# Pasos para optimizar SQL

Creamos un índice compuesto

```
create index i_cuenta_cliente_tipo_cuenta_ruc_entidad_bancaria  
on cuenta (cliente,tipo_cuenta,ruc_entidad_bancaria) ;
```

No olvidar luego crear un vacuum

```
vacuum full analyze cuenta;
```

```
create index i_cuenta_cliente_tipo_cuenta_ruc_entidad_bancaria on  
cuenta (cliente,tipo_cuenta,ruc_entidad_bancaria);
```

```
vacuum full analyze cuenta;
```

# Pasos para optimizar SQL

Ejecutamos el explain y evaluamos los resultados (paso 1)

```
28 EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON)
29 select p.cedula,p.nombres,ca.nombre,eb.nombre
30 from persona p
31 join cuenta c
32 on p.cedula=c.cliente
33 join catalogo ca
34 on ca.id_catalogo=c.tipo_cuenta
35 join entidad_bancaria eb
36 on eb.ruc=c.ruc_entidad_bancaria
37 where p.nombres='yoli'
```

Data Output Explain Messages Notifications

QUERY PLAN

json



1

[

# Pasos para optimizar SQL

Al copia el json al <https://explain.dalibo.com/plan#> evaluamos los resultados

## #4 Seq Scan

on public.cuenta as c

**Seq Scan Node** finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General IO & Buffers Output Workers Misc

Timing: 0,008ms | 14%

Rows: 1 (Planned: 626) | ↑ over estimated by 630 ×

Cost: 12,3 (Total: 12,3)

## #5 Bitmap Heap Scan

on public.cuenta as c

Never executed \$

**Bitmap Heap Scan Node** searches through the pages returned by the **Bitmap Index Scan** for relevant rows.

General IO & Buffers Output Workers Misc

Timing: 0ms | 0%

Rows: 0 (Planned: 2)

Cost: 4,56 (Total: 8,85)

## #6 Bitmap Index Scan

using

i\_cuenta\_cliente\_tipo\_cuenta\_ruc\_entidad\_bancaria

Never executed \$

**Bitmap Index Scan Node** uses a **Bitmap Index** (index which uses 1 bit per page) to find all relevant pages. Results of this node are fed to the **Bitmap Heap Scan**.

General IO & Buffers Output Workers Misc

Timing: 0ms | 0%

Rows: 0 (Planned: 2)

Cost: 4,29 (Total: 4,29)

# Ventajas de Índice

1. Evita lecturas secuenciales
2. Evitamos sobrecargar la CPU
3. Mejoramos la rapidez de la consultas

# Desventajas de Índice



1. Inserciones más lentas

## Pasos para optimizar SQL

Conclusión: Para optimizar la consulta se vio la necesidad de crear un índice simple en persona y un índice compuesto en la tabla en cuenta, para

# SQL: para optimizar la consulta

```
create index i_persona_nombre on persona(nombres)
vacuum full analyze persona; --Mantenimiento completo a la tabla y actualizacion de estadísticas de la tabla
(Tamaño de campos, cant de registros, que índices hay) útil para que el planificador de postgres pueda tomar la
mejor decisión
```

```
create index i_cuenta_cliente_tipo_cuenta_ruc_entidad_bancaria
on cuenta (cliente,tipo_cuenta,ruc_entidad_bancaria);
```

```
vacuum full analyze cuenta;
drop index i_cuenta_cliente_tipo_cuenta_ruc_entidad_bancaria
```

```
EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON)
```

```
select p.cedula,p.nombres,ca.nombre,eb.nombre
from persona p
join cuenta c
on p.cedula=c.cliente
join catalogo ca
on ca.id_catalogo=c.tipo_cuenta
join entidad_bancaria eb
on eb.ruc=c.ruc_entidad_bancaria
where p.nombres='yoli'
```

# SQL: para Ingresar Datos Masivamente

## 1. creamos una funcion numeros aleatorios

```
CREATE OR REPLACE FUNCTION random_between(low INT ,high INT)
  RETURNS INT AS
$$
BEGIN
  RETURN floor(random()* (high-low + 1) + low);
END;
$$ language 'plpgsql' STRICT;
```



# SQL: Para ingresar Datos Masivamente

## 2. creamos una función generar cadenas de letras tiras aleatorio

```
CREATE FUNCTION get_random_string(  
    IN string_length INTEGER,  
    IN possible_chars TEXT DEFAULT  
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'  
) RETURNS text  
LANGUAGE plpgsql  
AS $$  
DECLARE  
    output TEXT = "";  
    i INT4;  
    pos INT4;  
BEGIN  
    FOR i IN 1..string_length LOOP  
        pos := 1 + CAST( random() * ( LENGTH(possible_chars) - 1) AS INT4 );  
        output := output || substr(possible_chars, pos, 1);  
    END LOOP;  
    RETURN output;
```

# SQL: para probar las funciones realizadas

## 3. consultamos las funciones creadas

### **Numeros aleatorios**

```
select random_between(5000000,20000000)
```

### **Tira aleatoria**

```
Select get_random_string(  
    10,'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  
)
```

## SQL: Para ingresar Datos Masivamente

### 4. ingresamos datos de persona utilizando las funciones

```
insert into rol values(3,'auditor',true);insert into rol values(3,'auditor',true);
```

```
insert into rol values (1,'cliente',true)
```

```
insert into catalogo values(2,'Tipo_Cuenta','Corriente')
```

```
insert into catalogo values(1,'Tipo_Cuenta','ahorro')
```

```
insert into catalogo values(3,'estado_cuenta','activa');
```

```
insert into catalogo values(4,'estado_cuenta','inactiva');
```

```
insert into catalogo values(5,'estado_cuenta','bloqueada');
```

# SQL:Ingresar Datos Masivamente

## 5. ingresamos datos masivamente utilizando las funciones

```
insert into persona
```

```
select
```

```
    '|| random_between(5000000,20000000),'Nombre' ||  
random_between(17,50),'(02)' || random_between(1997,509  
9),'Direccion ' ||  
random_between(17,50),random_between(1,3)
```

```
from generate_series(0,100)
```

# SQL:Ingresar Datos Masivamente

## 5. ingresamos datos masivamente utilizando las funciones a la tabla cuenta

insert into cuenta

select '|| random\_between(1,5000),random\_between(1,2),  
random\_between(1,200),3,'999999','2020-01-01',500

from generate\_series(0,10)

# Tarea: ver video

Indice: <https://youtu.be/kq6H178T3Ww>

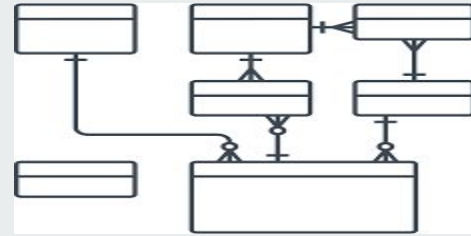
- 1- Ingresar masivamente datos a la base de datos Banco(transacción, detalle\_transaccion), usar funciones.
2. Realizar una consulta en la base de datos con join (mostrar las transacciones de la cuenta del sr. Tobias) (tomar captura de pantalla del tiempo que dure)
3. Aplicar indice a la tabla respectivas
4. Emitir conclusiones

# Trabajo final:

- 1- Validación: Transferencia de un cliente que no tenga suficiente saldo, no puede haber saldo negativo.
2. No se debe repetir el tipo de cuenta
3. Cuando ingreso un número de cuenta debe mostrar el nombre de la titular.
4. Al registrar una cuenta debe indicar el tipo de cuenta(ahorro, corriente) (tipo de cuenta es otra tabla).
5. Generar un reporte
6. Crear esquemas

# Clase 11

## Seguridad Usuarios y permisos



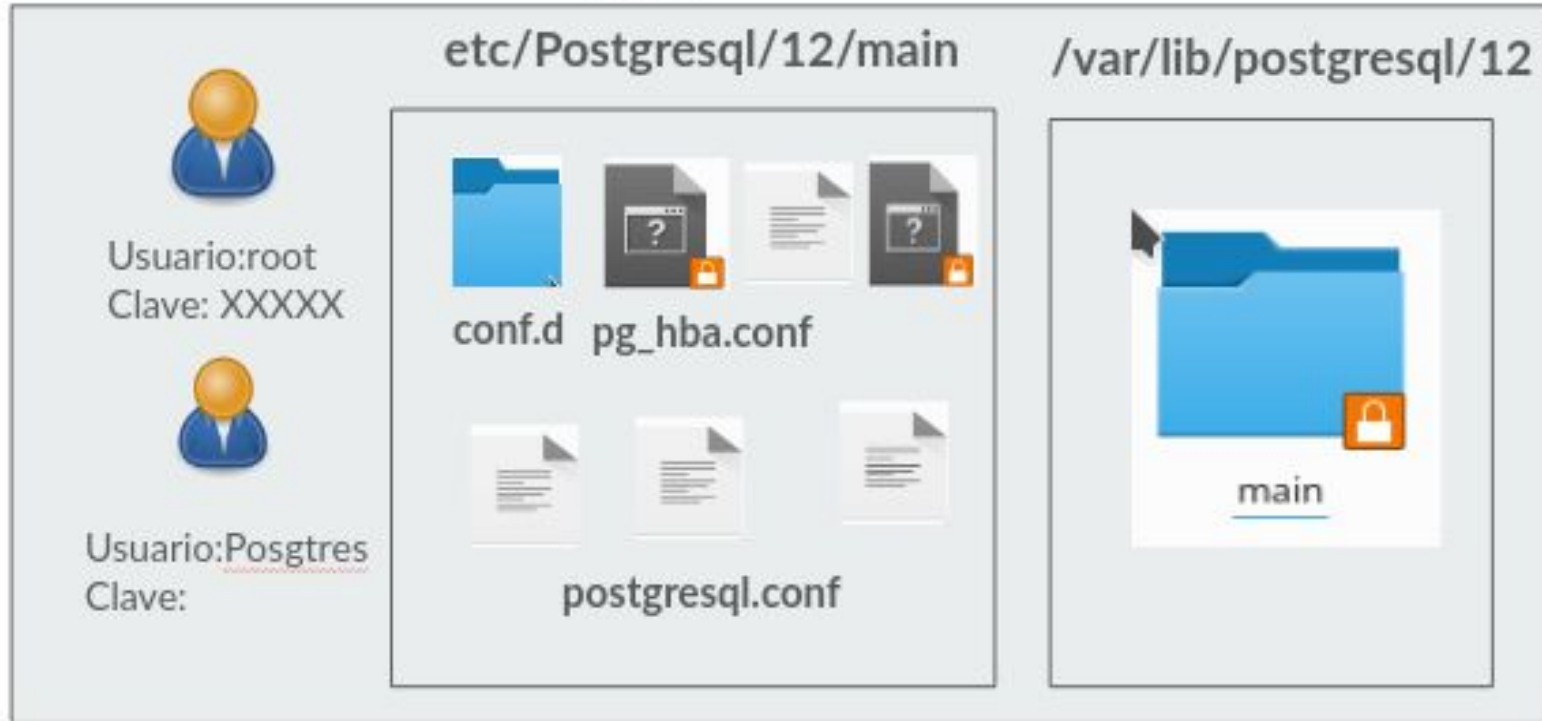


# Problemas en entornos de producción



# Configuración de postgres

**S.O.**



Ruta en Window\_  
C:\Program Files\PostgreSQL\13\data

# Configuración de postgres

- **postgresql.conf:** En este fichero podemos cambiar todos los parámetros de configuración que afectan al funcionamiento y al comportamiento de PostgreSQL en nuestra máquina.

Herramienta: PGtune

```
max_connections = 100
```

# Configuración de postgres

- **postgresql.conf:** Los cambios que realicemos en este fichero afectaran a todas las bases de datos que tengamos definidas en nuestro cluster PostgreSQL.

# Variables a Cambiar

**max\_connections:** Número máximo de clientes conectados a la vez a nuestras bases de datos. Deberíamos de incrementar este valor en proporción al número de clientes concurrentes en nuestro cluster PostgreSQL. Un buen valor para empezar es el 100:

```
max_connections = 100
```

## 2. Configuración uso recursos en Postgresql

- 1 #Conocer el servidor actual  
Memoria: `free -h`  
CPU: `lscpu`  
`htop`  
Disco: `sudo fdisk -l`
- 2 Consultar valores recomendados para propiedades de uso de recursos de postgresql.conf en pg tune
- 3 Modificar los valores recomendados por pg tune en un archivo postgresql.conf y habilitar `listen_addresses = '*'`
- 4 Reiniciar el servidor  
`sudo systemctl restart postgresql`

Error: Database connection failed

It is possible that the database is overloaded or otherwise not running properly.

The site administrator should also check that the database details have been correctly specified in config.php

Fuente: sistema: Ignug asistencia

Fecha: 27/07/2020, 12:48:00 pm

# Estimación de Usuarios de base de datos

Usuario de los aplicativos conectados simultaneamente					
Nombre Usuario	Total Usuario	Descripción	Nro de Usuarios conectados pos pc	Nro de Usuarios conectados pos Movil	Total
ignug_matricula	1600	Estudiantes+Docentes	1000	1000	2000
Eva_cecy	500	estudiantes y docentes	400	300	700
Eva_Yavirac	1000	estudiantes y docentes	800	500	1300
ignug_control_asistencia	100	docente	90	55	145
				Total de conexiones simultaneas	4145



## Configuración de postgres

**pg\_hba.conf:** Este fichero se utiliza para definir los diferentes tipos de accesos que un usuario tiene en el cluster.

Este fichero se utiliza para definir cómo, dónde y desde qué sitio un usuario puede utilizar nuestro cluster PostgreSQL. Todas las líneas que empiezan con el carácter `#` se interpretan como comentarios. El resto debe de tener el siguiente formato:

# Configuración de postgres

```
[Tipo de conexion][database][usuario][IP][Netmask][Tipo de autentificacion][opciones]
```

```
# TYPE      DATABASE    USER        ADDRESS      METHOD
# "local" is for Unix domain socket connections only
local      all         all          trust
host       ignug_db    ignug        172.17.0.3/16 md5
host       ignug_db    ignug        172.17.0.3/16 md5
host       all         postgres     172.17.0.3/16 md5
```

El tipo de conexion puede tener los siguientes valores, **local**, **host**, **all**, **hostssl** y **hostnossl**.

El tipo de método puede tener los siguientes valores, **trust**, **reject**, **md5**, **crypt**, **password**, **krb5**, **ident**, **pam** o **ldap**

# Enmaskamiento

Tabla de máscaras de red

Binario	Decimal	CIDR	Nº
11111111.11111111.11111111.11111111	255.255.255.255	/32	1
11111111.11111111.11111111.11111110	255.255.255.254	/31	2
11111111.11111111.11111111.11111100	255.255.255.252	/30	4
11111111.11111111.11111111.11111000	255.255.255.248	/29	8
11111111.11111111.11111111.11110000	255.255.255.240	/28	16
11111111.11111111.11111111.11100000	255.255.255.224	/27	32
11111111.11111111.11111111.11000000	255.255.255.192	/26	64
11111111.11111111.11111111.10000000	255.255.255.128	/25	128
11111111.11111111.11111111.00000000	255.255.255.0	/24	256
11111111.11111111.11111110.00000000	255.255.254.0	/23	512
11111111.11111111.11111100.00000000	255.255.252.0	/22	1024
11111111.11111111.11111000.00000000	255.255.248.0	/21	2048
11111111.11111111.11110000.00000000	255.255.240.0	/20	4096
11111111.11111111.11100000.00000000	255.255.224.0	/19	8192
11111111.11111111.11000000.00000000	255.255.192.0	/18	16384
11111111.11111111.10000000.00000000	255.255.128.0	/17	32768
11111111.11111111.00000000.00000000	255.255.0.0	/16	65536
11111111.11111110.00000000.00000000	255.254.0.0	/15	131072
11111111.11111100.00000000.00000000	255.252.0.0	/14	262144
11111111.11111000.00000000.00000000	255.248.0.0	/13	524288
11111111.11110000.00000000.00000000	255.240.0.0	/12	1048576
11111111.11100000.00000000.00000000	255.224.0.0	/11	2097152
11111111.11000000.00000000.00000000	255.192.0.0	/10	4194304
11111111.10000000.00000000.00000000	255.128.0.0	/9	8388608
11111111.00000000.00000000.00000000	255.0.0.0	/8	16777216
11111110.00000000.00000000.00000000	254.0.0.0	/7	33554432
11111100.00000000.00000000.00000000	252.0.0.0	/6	67108864
11111000.00000000.00000000.00000000	248.0.0.0	/5	134217728
11110000.00000000.00000000.00000000	240.0.0.0	/4	268435456
11100000.00000000.00000000.00000000	224.0.0.0	/3	536870912
11000000.00000000.00000000.00000000	192.0.0.0	/2	1073741824
10000000.00000000.00000000.00000000	128.0.0.0	/1	2147483648
00000000.00000000.00000000.00000000	0.0.0.0	/0	4294967296

# Ejemplo

**Ejemplo 1** .- Acceso por tcp/ip (red) a la base de datos **test001**, como usuario **test** desde el ordenador con **IP 10.0.0.100**, y método de autenticación **md5**:

```
host      test001      test  10.0.0.100  255.255.255.255  md5
```

Esta misma entrada se podría escribir también con la mascara de red en notación CIDR:

```
host      test001      test  10.0.0.100/32  md5
```

# Ejemplo

**Ejemplo 2** .- Acceso por tcp/ip (red) a la base de datos test001, como usuario test desde todos los ordenadores de la red 10.0.0.0, con mascara de red 255.255.255.0 (254 ordenadores en total) y metodo de autentificacion md5:

```
host      test001    test  10.0.0.0 255.255.255.0  md5
```

Esta misma entrada se podria escribir tambien con la mascara de red en notacion CIDR:

```
host      test001    test  10.0.0.0/24  md5
```

# Ejemplo

**Ejemplo 2** .- Acceso por tcp/ip (red) a la base de datos test001, como usuario test desde todos los ordenadores de la red 10.0.0.0, con mascara de red 255.255.255.0 (254 ordenadores en total) y metodo de autentificacion md5:

```
host      test001    test  10.0.0.0 255.255.255.0  md5
```

Esta misma entrada se podria escribir tambien con la mascara de red en notacion CIDR:

```
host      test001    test  10.0.0.0/24  md5
```

# Como crear y eliminar un usuario Y roles con contraseña

```
create role prueba2 password 'prueba2'  
drop role prueba2
```

```
create user prueba_user password 'prueba';  
ALTER USER prueba_user WITH PASSWORD 'prueba1';  
drop user prueba
```

# Al tratar de crear una conexión No es igual

Los Usuarios si se puede conectar

Los roles no se Pueden Conectar



# Como cambiar contraseña a un Rol y/o Usuario

## Cambiar la contraseña a un Usuario

```
ALTER USER prueba_user WITH encrypted PASSWORD 'prueba1';
```

## Cambiar contraseña a un Rol

```
alter role prueba_user with password 'prueba2'
```

# Permisos al usuario con ALTER ROLE

ALTER ROLE/USER, tiene varias opciones, entre ellas:

- SUPERUSER/NOSUPERUSER.
- CREATEDB/NOCREATEDB.
- CREATEROLE/NOCREATEROLE.
- CREATEUSER/NOCREATEUSER.
- LOGIN/NOLOGIN. Especificamos si será un rol o un usuario. El usuario tiene permisos para acceder a la base de datos a través de cualquier cliente, el rol no.
- PASSWORD.
- VALID UNTIL. Expiración de usuarios

# Permisos al usuario con ALTER ROLE

```
create user vinculacion_ddl PASSWORD '123' NOSUPERUSER
```

```
create user vinculacion_dml PASSWORD '123' NOSUPERUSER
```

```
create user ignug PASSWORD '123' SUPERUSER
```

# Otorgar privilegios sobre una base de datos

```
GRANT ALL PRIVILEGES ON DATABASE nombre_basedatos TO  
prueba2;
```

<https://www.postgresql.org/docs/12/sql-grant.html>

## Otorgar TODOS LOS privilegios sobre una base de datos

```
GRANT ALL PRIVILEGES ON SCHEMA  
vinculacion_sociedad to vinculacion_ddl;
```

## Ejemplo: Un usuario DDL

```
create user banco_ddl password '12345';
```

```
create user banco_ddl password '12345';
```

```
grant connect on database "banco_yogledis_V4" to banco_ddl;
```

```
grant all privileges on database "banco_yogledis_V4" to banco_ddl
```

```
grant all on all tables in schema transaccion to banco_ddl
```

```
revoke select,insert,update on all tables in schema transaccion from banco_ddl
```

---

```
--Probando
```

```
select * from transaccion.cuenta_prueba
```

```
create table transaccion.cuenta_prueba(cedula varchar(15) primary key)
```

```
drop table transaccion.cuenta_prueba
```

```
alter table transaccion.cuenta_prueba rename to transaccion.cuentapruoba1
```

## Ejemplo: Un usuario DML REALIZAR

--Usuario DML

create user banco\_dml password '12345'

revoke all privileges on database banco from banco\_dml

grant connect on database banco to banco\_dml;

grant usage on schema transaccion to banco\_dml

GRANT INSERT, UPDATE, DELETE, SELECT ON ALL TABLES IN SCHEMA transaccion TO banco\_dml;

**Ejemplo:** Crear, modificar, eliminar en el esquema vinculacionSociedad, consultar tablas

1. Creamos el usuario
2. Se le revoca todos los privilegios
3. Se le otorga los privilegios respectivo

# REVOCAR privilegios sobre una base de datos

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
```

revocar todos los privilegios

```
REVOKE ALL PRIVILEGES ON SCHEMA vinculacion_sociedad FROM  
vinculacion_ddl;
```



## Permisos al usuario con ALTER ROLE

```
create table ignug.estudiante(cedula character(20) primary key,nombre  
character(20),usuario_id character(20));
```

```
insert into ignug.estudiante values('15729191','yogledis')
```

```
create table authentication.usuario(login character(20) primary key,clave  
character(20));
```

```
insert into authentication.usuario values('yherrera','123')
```

# Otorgar Permisos de super usuario

**Otorgar Permisos a prueba de Superusuario**(estos permisos los otorga un super usuario postgresql)

```
alter user prueba with SUPERUSER
```

```
alter user prueba with NOSUPERUSER
```

# Pasos usuario

1. Creamos el usuario
2. Quitamos los Permisos
3. Otorgamos los permisos

## Tarea

Pasar todas las tablas de la base de datos banco al esquema transacciones y crear los siguientes usuarios con los siguientes permisos

Nombre Usuario	Permiso
banco_DDL	Crear, modificar, eliminar en el esquema transaccion
banco_DML	guarda, modifica, elimina datos del esquema transaccion

Probar que el usuario banco ddl no pueda manipular datos y el usuario banco dml no pueda ingresar realizar operaciones DDL

# Roles de la base de datos

- PostgreSQL gestiona los permisos de acceso a la base de datos utilizando el concepto de *roles* . Un rol puede considerarse como un usuario de la base de datos o un grupo de usuarios de la base de datos, dependiendo de cómo esté configurado el rol. Los roles pueden poseer objetos de base de datos (por ejemplo, tablas y funciones) y pueden asignar privilegios sobre esos objetos a otros roles para controlar quién tiene acceso a qué objetos. Además, es posible otorgar la *membresía* en un rol a otro rol, lo que permite que el rol de miembro use los privilegios asignados a otro rol.

El concepto de roles subsume los conceptos de " usuarios " y " grupos " . En las versiones de PostgreSQL anteriores a 8.1, los usuarios y grupos eran distintos tipos de entidades, pero ahora solo hay roles. Cualquier rol puede actuar como usuario, grupo o ambos.

# Seguridad

1. Plantea un posible ataque para cada uno de los principios de seguridad (integridad, confidencialidad, disponibilidad y autenticación) e indica una posible solución.

*-Integridad.*

*Ataque: robo de contraseña y edición de datos confidenciales.*

*Solución: mejor protección de las contraseñas.*

*-Confidencialidad.*

*Ataque: acceso a información privada a través de un troyano.*

*Solución: mejor detección de malware.*

*-Disponibilidad.*

*Ataque: ataque de denegación de servicio que impide acceder a un servidor.*

*Solución: servicios Cloud de protección anti DDoS que filtran las conexiones.*

*-Autenticación.*

*Ataque: uso de una contraseña generada automáticamente.*

*Solución: elevar criterios de seguridad de las contraseñas.*

# Seguridad

2. Indica, para los siguientes supuestos, qué principios de seguridad se están violando:
  - a. Destrucción de un elemento hardware  
*-Disponibilidad.*
  - b. Robo de un portátil con información de interés de la empresa  
*-Disponibilidad, confidencialidad e integridad.*
  - c. Robos de direcciones IP  
*-Autenticación.*
  - d. Escuchas electrónicas  
*-Confidencialidad.*
  - e. Modificación de los mensajes entre programas para variar su comportamiento  
*-Integridad.*
  - f. Deshabilitar los sistemas de administración de archivos  
*-Disponibilidad.*
  - g. Alterar la información que se transmite desde una base de datos  
*-Integridad.*
3. ¿Qué recomendaciones harías para evitar el acceso no autorizado a la información en una organización?

# Seguridad

*-Estudiar los posibles riesgos, impartir educación en materia de seguridad, controlar el acceso físico, utilizar firewall, estandarizar la seguridad mínima en las contraseñas, encriptar los envíos a través de la red y firmarlos digitalmente para verificar su procedencia.*

4. Busca información sobre los *sniffers* en Internet. ¿Qué son? ¿Qué utilidad tienen?

*-Un sniffer es un analizador de paquetes. Registra la información enviada desde un equipo. Es utilizada por administradores de red para mantener la seguridad en su redes, pero también por atacantes informáticos como método para robar información.*



## Conclusión

- Para el Entorno de producción se debe configurar el motor de base de datos.
- Antes de desarrollar un Sistema se debe tener definida y probada la arquitectura de la base de datos, considerando Integridad de datos, seguridad, rapidez en la generación de resultados

## Videos Recomendados

Vamos el siguiente

Video: <https://www.youtube.com/watch?v=sO3LEtxBsYg&feature=youtu>

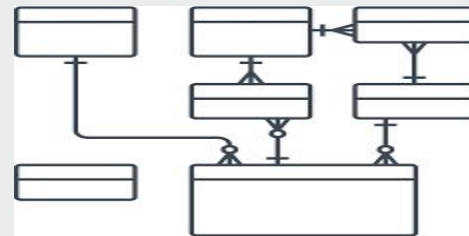
Vamos el siguiente Video: COMO CREAR USUARIO Y ASIGNAR ROL  
EN POSTGRESQL: [https://www.youtube.com/watch?v=Zjr9BQ\\_TXAo&feature=youtu.b](https://www.youtube.com/watch?v=Zjr9BQ_TXAo&feature=youtu.b)

Vamos el siguiente Video: Creación de Usuario en PostgreSQL:  
<https://youtu.be/WilhZjXBIL>

<https://youtu.be/WilhZjXBILE>

# Clase 8

## Reportes



# Herramientas Jasper Studio

## 1- Instalar herramienta de Reporte

<https://sourceforge.net/projects/jasperstudio/files/JaspersoftStudio-6.13.0/>



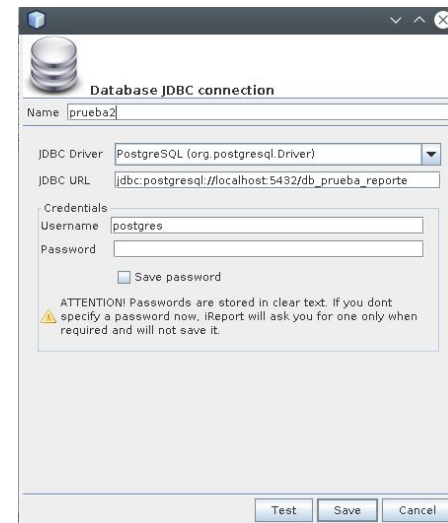
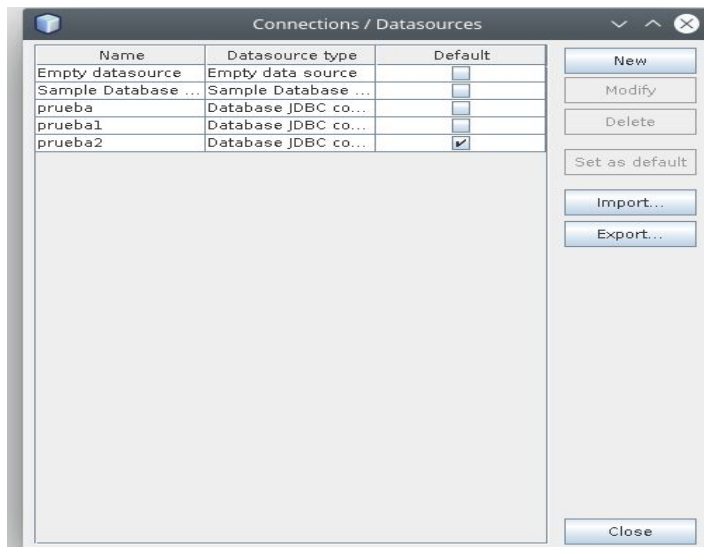
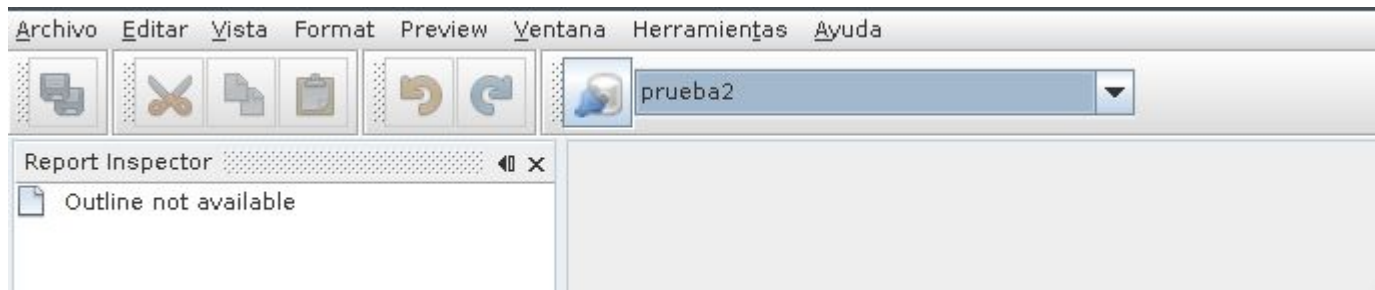
Jaspersoft® Studio CE

**v6.13.0**  
23 Jun 2020

Download

# Herramientas Jasper Reports

## 2- Crear una conexión a la base de datos



# Herramientas Jasper Server

## 1- Instalar herramienta de Reporte

<https://community.jaspersoft.com/project/jasperreports-server>

<https://community.jaspersoft.com/community-download>

<https://community.jaspersoft.com/community-download>

 Jaspersoft Community

PRODUCTS

SOLUTIONS

SERVICES

RESOURCES

Download Now

Answers

Exchange

Docs

Wiki

Planet

Tracker



# Resportes

Clase de reporte: <https://youtu.be/kVNLjSmSPWQ>

## Tarea

- Crear un reporte que permita visualizar todas las transacciones de un número de cuenta.
- El reporte se debe generar desde el sistema de banco que estamos trabajando a través de un botón



# Gracias.

