

Universidad de Ingeniería y Tecnología

FACULTAD DE COMPUTACIÓN

# ERP PARA INSTITUCIONES DE EDUCACIÓN PRIMARIA Y SECUNDARIA

HITO 1

AUTORES:

ESCOBAR NÚÑEZ, ALEJANDRO ISMAEL  
SILVA RIOS, ALESSANDRA VALERIA  
GALVEZ PACORI, JOSE GUILLERMO

PROFESOR:

RIOS OJEDA, BRENNER HUMBERTO

MAYO, 2024

# Contenidos

<b>1</b>	<b>Requisitos</b>	<b>3</b>
1.1	Introducción . . . . .	3
1.2	Descripción general del problema . . . . .	3
1.3	Necesidad y usos de la base de datos . . . . .	4
1.4	¿Cómo resuelve el problema hoy? . . . . .	4
1.4.1	¿Cómo se almacenan y procesan los datos? . . . . .	4
1.4.2	Flujo de datos . . . . .	4
1.5	Descripción detallada del sistema . . . . .	5
1.5.1	Objetos de información actuales . . . . .	5
1.5.2	Características y funcionalidades esperadas . . . . .	5
1.5.3	Tipos de usuarios existentes o necesarios . . . . .	5
1.5.4	Tipo de consulta y actualizaciones . . . . .	5
1.5.5	Tamaño estimado de la base de datos . . . . .	6
1.6	Objetivos del proyectos . . . . .	7
1.7	Referencias del proyecto . . . . .	8
1.8	Eventualidades . . . . .	8
1.8.1	Problemas que pudieran encontrarse en el proyecto . . . . .	8
1.8.2	Límites y alcances del proyecto . . . . .	8
<b>2</b>	<b>Modelo Entidad-Relación</b>	<b>9</b>
2.1	Reglas semánticas . . . . .	9
2.2	Modelo Entidad-Relación . . . . .	10
2.3	Especificaciones y consideraciones sobre el modelo . . . . .	11
2.3.1	Entidad Institucion . . . . .	11
2.3.2	Entidad Persona . . . . .	11
2.3.3	Entidad Alumno . . . . .	11
2.3.4	Entidad Apoderado . . . . .	11
2.3.5	Entidad Colaborador . . . . .	11
2.3.6	Entidad Profesor . . . . .	12
2.3.7	Entidad Secretario . . . . .	12
2.3.8	Entidad Director . . . . .	12
2.3.9	Entidad Consejero . . . . .	12
2.3.10	Entidad Tutor . . . . .	12
2.3.11	Entidad Sede . . . . .	13
2.3.12	Entidad Salon . . . . .	13
2.3.13	Entidad Grado . . . . .	13
2.3.14	Entidad Curso . . . . .	13
2.3.15	Entidad Matrícula . . . . .	13
<b>3</b>	<b>Modelo Relacional</b>	<b>14</b>
3.1	Modelo Relacional . . . . .	14
3.2	Especificaciones de transformación . . . . .	15
3.2.1	Entidades . . . . .	15
3.2.2	Entidades débiles . . . . .	15

3.2.3	Entidades superclases y subclases . . . . .	15
3.2.4	Relaciones binarias . . . . .	16
3.2.5	Relaciones ternarias . . . . .	16
3.3	Diccionario de datos . . . . .	16
<b>4</b>	<b>Implementación de la base de datos</b>	<b>23</b>
4.1	Creación de tablas en PostgreSQL . . . . .	23
4.2	Creación de triggers en PostgreSQL . . . . .	25
4.3	Carga de datos . . . . .	31
4.4	Simulación de datos faltantes . . . . .	31
<b>5</b>	<b>Optimización y experimentación</b>	<b>32</b>
5.1	Consultas SQL para el experimento . . . . .	32
5.1.1	Descripción del tipo de consultas seleccionadas . . . . .	32
5.1.2	Implementación de consultas en SQL . . . . .	33
5.2	Metodología del experimento . . . . .	35
5.3	Optimización de consultas . . . . .	35
5.3.1	Planes de índices para la primera consulta . . . . .	35
5.3.2	Planes de índices para la segunda consulta . . . . .	40
5.3.3	Planes de índices para la tercera consulta . . . . .	45
5.4	Plataforma de pruebas . . . . .	51
5.5	Medición de tiempos . . . . .	51
5.5.1	Sin índices . . . . .	51
5.5.2	Con índices . . . . .	51
5.6	Resultados . . . . .	51
5.6.1	Consulta 1 . . . . .	51
5.6.2	Consulta 2 . . . . .	51
5.6.3	Consulta 3 . . . . .	51
<b>6</b>	<b>Análisis y discusión</b>	<b>51</b>
<b>7</b>	<b>Conclusiones</b>	<b>51</b>
<b>8</b>	<b>Anexos</b>	<b>51</b>
8.1	Modelo Físico . . . . .	51
8.2	Videos de experimentación . . . . .	51
8.2.1	Consulta 1 . . . . .	51
8.2.2	Consulta 2 . . . . .	51
8.2.3	Consulta 3 . . . . .	51
8.2.4	Consulta 4 . . . . .	51
8.3	Pregunta extra . . . . .	51

# 1 Requisitos

## 1.1 Introducción

En el panorama educativo contemporáneo, las instituciones escolares representan no solo espacios de aprendizaje, sino también nodos vitales en la estructura social y cultural de una comunidad. La gestión eficiente de estos centros educativos no solo implica la impartición de conocimientos, sino también la administración fluida de una vasta cantidad de datos que abarcan desde la información personal de los estudiantes hasta los registros académicos y administrativos. En este contexto, surge la necesidad imperante de contar con sistemas de gestión de datos eficaces y estructurados que permitan a las instituciones educativas optimizar sus procesos internos y brindar un servicio de calidad.

En respuesta a esta demanda, el presente informe aborda la problemática específica que enfrentan los colegios en cuanto a la gestión de datos. En un mundo cada vez más digitalizado, donde la información es un recurso invaluable, la falta de un sistema adecuado para almacenar, organizar y acceder a los datos relevantes puede generar ineficiencias significativas en la operatividad diaria de las instituciones educativas. Es en este contexto que se plantea la necesidad de desarrollar una base de datos estructurada y accesible que facilite la gestión integral de los datos necesarios para el funcionamiento óptimo de un colegio.

El proyecto en cuestión se centra en la creación de un modelo de base de datos diseñado específicamente para abordar los desafíos mencionados. A través de la implementación de diversas tablas y relaciones, se busca proporcionar una solución integral que permita a los colegios gestionar de manera eficiente información crucial fundamental para su operación. Este informe detalla el proceso de diseño, desarrollo e implementación de la base de datos, así como su potencial impacto en la optimización de la gestión escolar.

## 1.2 Descripción general del problema

Manejar los recursos, las finanzas, los inventarios, y las demás áreas necesarias para el buen funcionamiento de una empresa es un problema constante para cualquier compañía incipiente o veterana. Por ello, aunque existan distintos sistemas ERP de paga y open source que han dado una solución tecnológica a esta problemática, estos siguen siendo de difícil acceso para las PYMES de nuestro país debido al costo —en caso de los sistemas ERP de pago (SAP, Oracle Fusion Cloud ERP, Microsoft Dynamics 365, etc.)— o muy complejas de implementar y entender —en caso de los sistemas ERP open source (Odoo, ERPNext, Dolibarr, etc.). Las instituciones pequeñas y medianas de educación primaria y secundaria no son ajenas a esta necesidad, en consecuencia, distintas formas más o menos tecnológicas han sido implementadas para satisfacerla: desde anotaciones a lápiz y papel hasta tablas y tablas de Excel.

### 1.3 Necesidad y usos de la base de datos

Debido a la problemática antes descrita, es necesario proporcionar una base de datos para un sistema ERP open source simple y eficiente enfocado a los colegios de primaria y secundaria. Esta base de datos manejaría las entidades clave de todo colegio: alumnos, profesores, cursos, grados, salones, directores, secretarías, consejeros, etc; además, estaría preparada para soportar la incorporación de nuevas sedes.

Un aspecto medular a todas las instituciones educativas son las matrículas, por ello, esta base de datos permitiría a los colegios capturar y guardar la información pertinente a cada matrícula: desde el alumno y su apoderado hasta la secretaria que realiza dicha matrícula.

Por último, aunque el motivo principal de esta base de datos sea la de servir a un ERP, fácilmente podría adaptarse a una página web informativa sobre las sedes, profesores, grados y cursos que brinda la institución educativa.

### 1.4 ¿Cómo resuelve el problema hoy?

#### 1.4.1 ¿Cómo se almacenan y procesan los datos?

Hoy en día, en muchas escuelas, los datos se almacenan de manera dispersa y poco estructurada. Por lo general, se utilizan métodos tradicionales como hojas de cálculo, archivos físicos y sistemas informáticos fragmentados para gestionar la información. Los registros de estudiantes, personal docente, calificaciones y otros datos relevantes suelen estar dispersos en diferentes sistemas y formatos, lo que dificulta su acceso y gestión eficiente. Esta falta de centralización y estandarización puede generar problemas de integridad de datos y dificulta la generación de informes y análisis exhaustivos para la toma de decisiones fundamentadas.

#### 1.4.2 Flujo de datos

Por lo general, los datos se recopilan y actualizan de forma independiente en cada sistema o plataforma utilizada en la escuela. Por ejemplo, los datos de los estudiantes pueden ingresarse en un sistema de gestión académica separado, mientras que la información del personal docente puede almacenarse en otro sistema diferente. Esta falta de integración dificulta la sincronización y actualización de datos en tiempo real, lo que a menudo resulta en inconsistencias y redundancias en la información. Además, el intercambio de datos entre diferentes sistemas puede requerir procesos manuales de exportación e importación, lo que aumenta el riesgo de errores y demoras en la disponibilidad de la información actualizada. En resumen, el flujo de datos en este contexto tiende a ser fragmentado y poco eficiente, lo que limita la capacidad de la escuela para gestionar de manera efectiva su información.

## **1.5 Descripción detallada del sistema**

### **1.5.1 Objetos de información actuales**

### **1.5.2 Características y funcionalidades esperadas**

- Administrar toda la información del colegio en una única plataforma, desde los datos de los estudiantes y el personal hasta los detalles de la infraestructura y los recursos académicos.
- Fácil de usar por los administradores, profesores, y personal administrativo del colegio, facilitando la navegación y la recuperación de información de manera eficiente.
- Automatizar tareas rutinarias y repetitivas, lo que ayuda a reducir la carga administrativa y minimizar errores humanos.
- Escalable y adaptable, permitiendo la incorporación de nuevas funcionalidades y el manejo de un creciente volumen de datos a medida que la institución crece.

### **1.5.3 Tipos de usuarios existentes o necesarios**

### **1.5.4 Tipo de consulta y actualizaciones**

- Principales consultas:
  - ¿Qué sedes existen?
  - ¿Quién es el director de cada sede?
  - ¿Qué profesores existen por sede?
  - ¿Qué tutor hay en cada salón de cada sede?
  - ¿Qué secretarios existen en cada sede?
  - ¿Qué consejeros existen en cada sede?
  - ¿Qué sede tiene más o menos alumnos?
  - ¿Cuántos alumnos han sido matriculados en determinado año en una sede o en todas las sedes?
  - ¿Quién es el apoderado de cada alumno?
  - ¿Cuántas matrículas tiene una sede por año?
  - ¿Qué secretaria registra más matrículas en cada sede y en todas las sedes?
  - Obtener la lista de grados con los cursos asignados y sus respectivos profesores.
- Principales actualizaciones:
  - Actualización manual de la información general de la institución educativa.
  - Actualización manual de los profesores, cursos, grados, sedes, alumnos, apoderados, directores, consejeros, secretarios y salones.

### 1.5.5 Tamaño estimado de la base de datos

Nombre de la tabla	Longitud de atributos en Bytes	Longitud del registro en Bytes
Alumno	$8 + 50 + 4 + 4 + 8$	74
Apoderado	$8 + 15$	23
Colaborador	$8 + 8 + 20 + 15 + 4$	55
Consejero	$8 + 4$	12
Curso	$50 + 4$	54
Grado	$50 + 4$	54
InformacionInstitucion	$11 + 1000 + 100 + 4 + 255 + 150$	1520
Matricula	$8 + 4 + 4 + 4 + 8$	28
Persona	$8 + 100 + 50 + 50 + 4 + 1 + 100$	313
ProfesorCursoGrado	$8 + 4 + 4 + 4$	20
ProfesorSede	$8 + 4$	12
Profesor	8	8
Salon	$4 + 4 + 50 + 8 + 4$	70
Secretario	$8 + 4$	12
Sede	$4 + 8 + 8 + 255 + 8 + 8$	291
Tutor	$8 + 4$	12

Table 1: Estimación del tamaño de la base de datos

Nombre de la Tabla	Tamaño en Bytes	Número de Datos Estimados	Tamaño Total en Bytes
Curso	54	100	5400
Grado	54	30	1620
Salon	70	50	3500
InformacionInstitucion	1520	1	1520
Sede	291	5	1455
Profesor	8	150	1200
Secretario	12	10	120
Tutor	12	20	240
Consejero	12	15	180

Table 2: Estimación del tamaño de las tablas fijas

Nombre de la Tabla	Tamaño en Bytes	Crecimiento de Datos por Día	Crecimiento de Datos Anual	Tamaño Anual de Bytes
Alumno	74	5	1825	135050
Matricula	28	10	3650	102200
Persona	313	5	1825	571125
Apoderado	23	2	730	16790
Colaborador	55	1	365	20075
ProfesorCursoGrado	20	3	1095	21900
ProfesorSede	12	1	365	4380

Table 3: Estimación del crecimiento de las tablas cambiantes

## 1.6 Objetivos del proyectos

- Crear una base de datos unificada que consolide toda la información relevante del colegio, incluyendo datos de estudiantes, personal, y recursos, eliminando la fragmentación y redundancia de datos.
- Facilitar el acceso a la información de manera rápida y eficiente para todos los usuarios autorizados, permitiendo consultas y generación de informes apropiados para la toma de decisiones.



- Reducir la carga administrativa mediante la automatización de tareas rutinarias aumentando la eficiencia y disminuyendo el riesgo de errores manuales.

## 1.7 Referencias del proyecto

Las referencias para este proyecto se basan en las experiencias de los integrantes del equipo en colegios pequeños, donde hemos podido observar de primera mano los desafíos relacionados con la gestión manual de datos. En estos entornos, hemos sido testigos de cómo la información crítica, desde la matriculación de estudiantes hasta el seguimiento académico, se registra en formatos físicos como archivos escritos o se gestiona mediante hojas de cálculo de Excel. Esta práctica, aunque común, a menudo conlleva dificultades en la organización, acceso y actualización de los datos, lo que puede resultar en errores y retrasos en la toma de decisiones. Esta experiencia práctica nos ha inspirado a desarrollar una solución que aborde estas necesidades específicas, basada en una comprensión profunda de los desafíos enfrentados por las instituciones educativas en la gestión de datos en la actualidad.

## 1.8 Eventualidades

### 1.8.1 Problemas que pudieran encontrarse en el proyecto

- Un potencial problema sería matricular a un alumno para determinada sede, pero que las vacantes para esa sede se hayan agotado (en el contexto de nuestro proyecto, nos referíamos al aforo de determinada aula que corresponde a determinado grado).

### 1.8.2 Límites y alcances del proyecto

- Límites: Nuestro proyecto, a diferencia de un ERP robusto, no contempla un manejo completo del área de RR.HH., sin embargo, contemplamos una solución simple: atributos como sueldo o cci para los colaboradores. Además, tampoco abordamos funcionalidades adicionales como integración con sistemas externos, gestión avanzada de recursos financieros, o herramientas de análisis de datos complejos.
- Alcances: Nuestro proyecto cubre una amplia gama de funcionalidades necesarias para gestionar una institución educativa, desde la administración de estudiantes y personal hasta la gestión de cursos y salones. Además, proporcionamos una base sólida para integrar todos los datos relevantes de una institución en un solo lugar, lo que facilita el acceso y la gestión de la información.

## 2 Modelo Entidad-Relación

### 2.1 Reglas semánticas

- Una institución tiene descripción, banner, nombre, fue fundado en cierta fecha y por alguien, y tiene RUC como identificador.
- Una institución tiene al menos una sede, y en ella trabajan colaboradores y se dictan clases.
- Una persona tiene nombres, apellidos, fecha de nacimiento, sexo, email y es identificada por su DNI.
- Un alumno es una persona, puede ser matriculado por solo un apoderado y estudia en un solo salón.
- Un apoderado es una persona, tiene número de celular y puede matricular a uno o más alumnos.
- Un colaborador es una persona, tiene sueldo por hora, cci, numero de celular, horas semanales de trabajo y necesitamos saber si está activo o no.
- Un profesor es un colaborador y trabaja en una o más sedes enseñando uno o más cursos en uno o más grados en cierto periodo académico.
- Un secretario es un colaborador y trabaja en una sola sede.
- Un director es un colaborador y dirige una sola sede.
- Un consejero es un colaborador y trabaja en una sola sede.
- Un tutor es un colaborador, trabaja en una sola sede y se le asigna vigilar un solo salón.
- Una sede tiene un id como identificador, una dirección y sus respectivas coordenadas, además de la fecha de su construcción, es dirigida por un solo director y en ella trabajan, uno o más consejeros, uno o más secretarios, uno o más tutores y uno o más profesores, y tiene uno o más salones.
- Un salón tiene un nombre de sección como llave parcial, cuenta con un número para el aforo, además, pertenece a una sola sede, es vigilado por un tutor, en él estudian muchos alumnos y en él se dictan clases de un solo grado.
- Un grado está identificado por un id, tiene un nombre, las clases de un grado son dictadas en uno o más salones por un profesor en cierto periodo académico y este contiene muchos cursos.
- Un curso está identificado por un id, posee un nombre, está dictado por un profesor y está contenido en uno o más grados.
- Una matrícula es realizada en cierto año por un apoderado al darle los datos de un alumno a un secretario para que lo registre en un grado y una sede.

## 2.2 Modelo Entidad-Relación

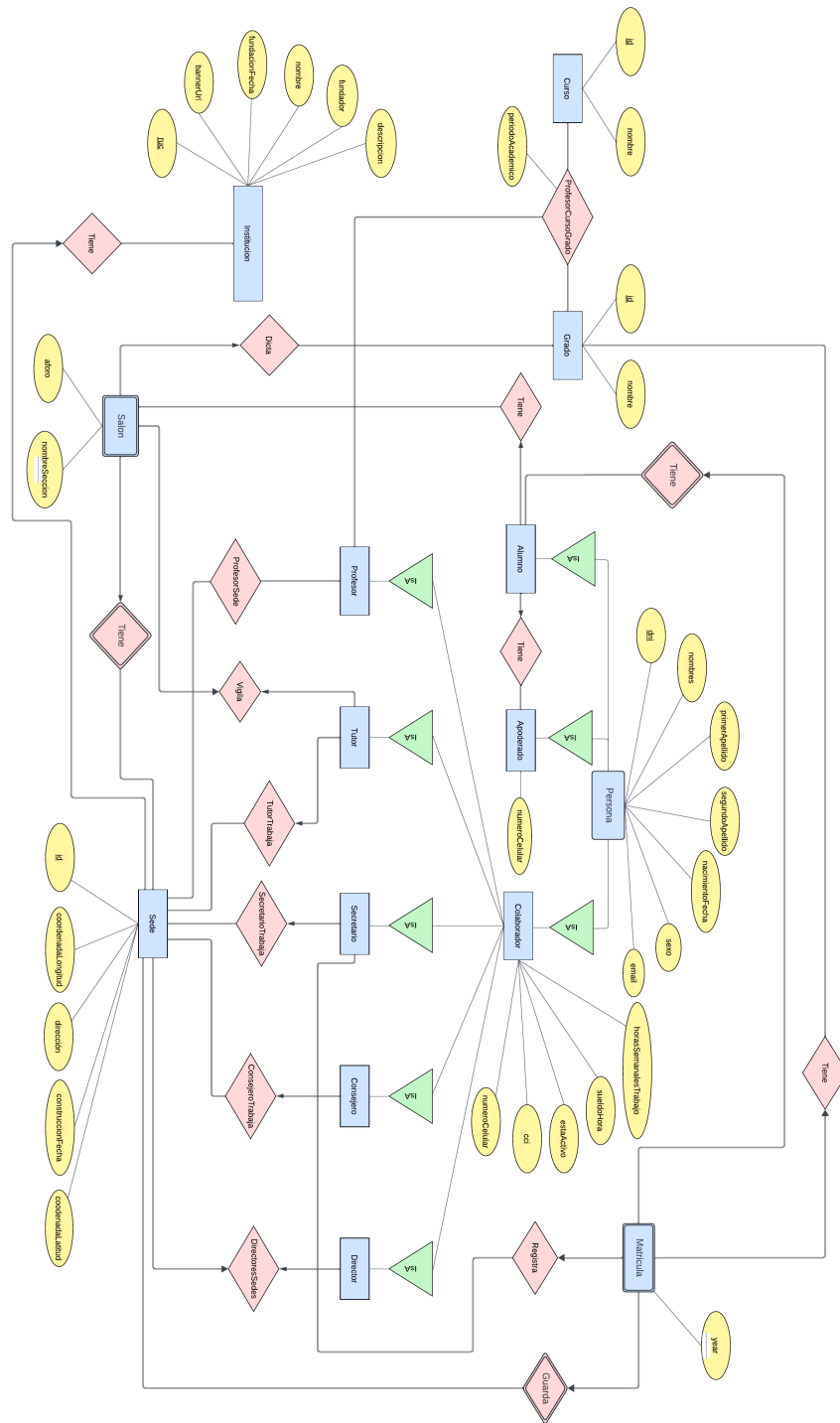


Figure 1: Modelo Entidad-Relación

## **2.3 Especificaciones y consideraciones sobre el modelo**

### **2.3.1 Entidad Institucion**

- Especificaciones: Almacena la información más importante de cada institución: descripción, banner, nombre, fecha de fundación, fundador y el RUC como llave primaria.
- Consideraciones: El RUC es seleccionado como la llave primaria ya que es un identificador único y oficial para cada institución. Esto facilita la gestión administrativa y legal de la información de la institución. Esta entidad no posee relaciones con ninguna otra entidad, ya que solo almacenará una tupla la cual tendrá la información antes descrita.

### **2.3.2 Entidad Persona**

- Especificaciones: Almacena la información más importante de cada individuo, incluyendo DNI (como llave primaria), nombres, apellidos, fecha de nacimiento, sexo y email.
- Consideraciones: El DNI es usado como llave primaria, proporcionando un medio único y eficiente para identificar a cada persona. Esta entidad actúa como una superclase que hereda a las subclases Alumno, Apoderado y Colaborador, permitiendo solapamiento y cobertura total.

### **2.3.3 Entidad Alumno**

- Especificaciones: Subclase de Persona. Está vinculado a un único salón y es matriculado por un solo apoderado.
- Consideraciones: Hereda el DNI de Persona como llave primaria. La relación exclusiva con un apoderado y un salón justifica el mantenimiento de estas conexiones a través de claves foráneas, lo cual asegura integridad referencial y facilita consultas relacionales.

### **2.3.4 Entidad Apoderado**

- Especificaciones: Subclase de Persona. Posee un número de celular adicional y puede matricular a uno o más alumnos.
- Consideraciones: Utiliza el DNI de Persona como llave primaria. La capacidad de vincularse con múltiples alumnos refuerza su rol en el proceso educativo y administrativo, y la integridad referencial se mantiene a través de relaciones definidas en la base de datos.

### **2.3.5 Entidad Colaborador**

- Especificaciones: Subclase de Persona. Incluye sueldo por hora, CCI, número de celular, horas semanales de trabajo y un atributo para saber si está activo o no.

- Consideraciones: Continúa usando el DNI como llave primaria. Actúa como superclase para Profesor, Secretario, Director, Consejero, y Tutor, facilitando la implementación de políticas de empleo y administración dentro del colegio.

#### **2.3.6 Entidad Profesor**

- Especificaciones: Subclase de Colaborador. Puede enseñar en una o más sedes.
- Consideraciones: Mantiene el DNI como llave primaria. La flexibilidad de enseñar en múltiples sedes es gestionada mediante una relación muchos a muchos con Sede, lo cual permite una programación eficiente.

#### **2.3.7 Entidad Secretario**

- Especificaciones: Secretario es una subclase de Colaborador y trabaja en una sola sede.
- Consideraciones: El DNI es la llave primaria. Establece una relación muchos a uno con Sede la cual es justificada a través de una clave foránea, lo que garantiza que cada sede tenga un único secretario asociado.

#### **2.3.8 Entidad Director**

- Especificaciones: Director es una subclase de Colaborador y dirige una sola sede.
- Consideraciones: El DNI es la llave primaria y sedeId como llave foránea. La exclusividad de la relación uno a uno con Sede asegura un manejo claro de responsabilidades administrativas.

#### **2.3.9 Entidad Consejero**

- Especificaciones: Consejero es una subclase de Colaborador y trabaja en una sede.
- Consideraciones: Utiliza el DNI como llave primaria. La relación muchos a uno con Sede facilita la asignación de responsabilidades y roles dentro del colegio y realiza la conexión a través de una llave foránea.

#### **2.3.10 Entidad Tutor**

- Especificaciones: Tutor es una subclase de Colaborador, trabaja en una sede y se le asigna un salón.
- Consideraciones: El DNI como llave primaria, nombreSeccion y sedeId como llaves foráneas, además su relación específica con un salón ayuda a mantener un control efectivo sobre el ambiente educativo.

#### **2.3.11 Entidad Sede**

- Especificaciones: Sede tiene un identificador único (ID), dirección, coordenadas y fecha de construcción. Está dirigida por un Director.
- Consideraciones: El ID como llave primaria y el RUC de la institución como llave foránea.

#### **2.3.12 Entidad Salon**

- Especificaciones: Salón tiene como identificador nombreSeccion, tiene aforo, pertenece a una sede, en este se dictan clases de un grado y es supervisado por un Tutor.
- Consideraciones: Tiene una llave primaria compuesta por los atributos nombreSeccion y sedeId, además de tener a gradoId como llave foránea lo cual permite un manejo detallado de los espacios físicos. Es considerada una entidad débil, ya que no podría existir un salón si es que no existe una sede al igual que no tendría sentido tener un salón sin que ninguna clase de algún grado se realice en él.

#### **2.3.13 Entidad Grado**

- Especificaciones: Grado está identificado por un ID y tiene un nombre. Contiene varios cursos y se enseña en varios salones.
- Consideraciones: El ID como llave primaria facilita la organización de los programas educativos. Se relaciona con Profesor, Curso y Salón para estructurar el currículo.

#### **2.3.14 Entidad Curso**

- Especificaciones: Curso tiene un ID y un nombre, y está contenido en uno o más grados.
- Consideraciones: El ID como llave primaria es adecuado para la gestión curricular. La relación con Grado y Profesor permite múltiples configuraciones pedagógicas.

#### **2.3.15 Entidad Matrícula**

- Especificaciones: Involucra Alumno, Apoderado, Grado, Sede y es realizada por un Secretario. Usa una clave primaria compuesta de alumnoDni, year, y sedeId.
- Consideraciones: La clave primaria compuesta asegura una identificación única de cada registro, facilitando la administración y el seguimiento académico. Es una entidad débil, ya que para existir necesita que un apoderado matricule a un alumno con la ayuda de un secretario en cierto grado en cierta sede.

## 3 Modelo Relacional

### 3.1 Modelo Relacional

- Institucion (ruc, descripcion, fundador, fundacionFecha, bannerUrl, nombre)
- Persona (dni, nombres, primerApellido, segundoApellido, nacimientoFecha, sexo, email)
- Apoderado (Persona.dni, numeroCelular)
- Alumno (Persona.dni, Salon.nombreSeccion, Sede.id, Apoderado.dni)
- Colaborador (Persona.dni, sueldoHora, cci, numeroCelular, horasSemanalesTrabajo, estaActivo)
- Secretario (Colaborador.dni, Sede.id)
- Consejero (Colaborador.dni, Sede.id)
- Director (Colaborador.dni, Sede.id)
  - Relación One To One (1:1) con la entidad Sede, usamos el constraint UNIQUE en el traspaso hacia las tablas SQL, para asegurar que un Director solo pueda ser asignado a una sede.
- Tutor (Colaborador.dni, Salon.nombreSeccion, Sede.id)
  - Relación One To One (1:1) con la entidad Salon, usamos el constraint UNIQUE en el traspaso hacia las tablas SQL, para asegurar que un Tutor solo pueda ser asignado a un salon.
- Profesor (Colaborador.dni)
- ProfesorSede (Profesor.dni, Sede.id)
- Sede (id, coordenadaLongitud, coordenadaLatitud, direccion, construccionFecha, Institucion.ruc)
- Grado (id, nombre)
- Curso (id, nombre)
- ProfesorCursoGrado (Curso.id, Grado.id, Profesor.dni, periodoAcademico)
- Salon (nombreSeccion, Sede.id, Grado.id, Tutor.dni, aforo)
- Matricula (year, Alumno.dni, Sede.id, Grado.id, Secretario.dni)

## 3.2 Especificaciones de transformación

### 3.2.1 Entidades

- **Curso:** Se transforma en la tabla Curso con id como llave primaria. Cada curso tiene un nombre único. No hay dependencia directa con otras tablas a nivel de llave primaria.
- **Grado:** Se convierte en la tabla Grado con id como llave primaria, y nombre como atributo. Los grados organizan los cursos y se vinculan directamente con varios salones.
- **Sede:** Se transforma en la tabla Sede con id como llave primaria, además de tener al RUC de la institución como llave foránea. Incluye atributos como direccion, coordenadaLongitud, coordenadaLatitud, y construccionFecha. Representa una ubicación física donde se imparten cursos y trabajan los colaboradores.
- **Institucion:** Se convierte en la tabla Institucion con ruc como llave primaria. Incluye descripcion, fundador, fundacionFecha, bannerUrl, nombre. Esta entidad encapsula los datos fundamentales de la institución educativa.

### 3.2.2 Entidades débiles

- **Salon:** Se convierte en la tabla Salon con una llave primaria compuesta por nombreSeccion y sedeId. Dependiente de Sede, reflejando que cada salón está ubicado en una sede específica. Atributos incluyen aforo, gradoId como foreign key.
- **Matricula:** La tabla Matricula define su llave primaria compuesta por alumnoDni, sedeId, y year, lo que refleja que un alumno se puede matricular en una sede específica cada año. Las claves foráneas incluyen gradoId y secretarioDni. gradoId vincula la matrícula al grado específico al cual el alumno está inscrito, facilitando la organización académica. secretarioDni conecta cada matrícula al secretario que procesó la inscripción, integrando la administración del proceso.

### 3.2.3 Entidades superclases y subclases

- **Persona:** Superclase que se transforma en la tabla Persona con dni como llave primaria. Todos los individuos (alumnos, apoderados, colaboradores) se derivan de esta tabla, heredando dni y demás atributos personales.
- **Alumno, Apoderado, Colaborador:** Subclases de Persona. Cada una con sus respectivas tablas donde dni actúa como clave foránea y primaria. Alumno incluye nombreSeccion, sedeId y apoderadoDni, mostrando la dependencia y relaciones con otras entidades.



- **Profesor, Tutor, Secretario, Consejero, Director:** Subclases de Colaborador, cada una con roles y responsabilidades definidos, vinculados a sedes y otros elementos estructurales de la institución.

#### 3.2.4 Relaciones binarias

- **Salon y Sede:** Cada salón pertenece a una sede, representando una relación de 1 a n, donde cada sede puede contener varios salones.
- **Alumno y Salon:** Relación de n a 1, cada alumno está asignado a un salón específico.
- **Grado y Salon:** Relación de 1 a n, cada grado se imparte en varios salones, mostrando que un salón puede ser utilizado para diferentes grados dependiendo del horario y necesidad académica.
- **Profesor y Sede:** Esta relación binaria indica cómo los profesores están asignados a sedes específicas. La multiplicidad muestra que un profesor puede estar asignado a varias sedes, y cada sede puede tener múltiples profesores.

#### 3.2.5 Relaciones ternarias

- **ProfesorCursoGrado:** Esta relación muestra que los cursos son ofrecidos en varios grados por diferentes profesores. La multiplicidad aquí refleja que un curso puede ser impartido en varios grados y que múltiples profesores pueden enseñar el mismo curso en diferentes grados.

### 3.3 Diccionario de datos

Nombre del campo	Tipo de dato	PK	FK	Descripción
Persona.dni	CHAR(8)	X	X	DNI del alumno.
Salon.nombreSeccion	VARCHAR(50)			Nombre de la sección.
Sede.id	INT		X	ID de la sede.
Apoderado.dni	CHAR(8)		X	DNI del apoderado.

Table 4: Alumno

Nombre del campo	Tipo de dato	PK	FK	Descripción
Persona.dni	CHAR(8)	X	X	DNI de la persona que es apoderado.
numeroCelular	VARCHAR(15)			Número de celular del apoderado.

Table 5: Apoderado

Nombre del campo	Tipo de dato	PK	FK	Descripción
Persona.dni	CHAR(8)	X	X	DNI del colaborador.
sueldoHora	FLOAT			Sueldo por hora del colaborador.
cci	CHAR(20)			Código de cuenta interbancaria.
numeroCelular	VARCHAR(15)			Número de celular del colaborador.
horasSemanalesTrabajo	INT			Horas semanales de trabajo.
estaActivo	BOOLEAN			Indica si el colaborador está activo o no.

Table 6: Colaborador

Nombre del campo	Tipo de dato	PK	FK	Descripción
Colaborador.dni	CHAR(8)	X	X	DNI del consejero, que es un tipo de colaborador.
Sede.id	INT		X	ID de la sede donde trabaja el consejero.

Table 7: Consejero

Nombre del campo	Tipo de dato	PK	FK	Descripción
id	INT	X		ID del curso.
nombre	VARCHAR(50)			Nombre del curso.

Table 8: Curso

Nombre del campo	Tipo de dato	PK	FK	Descripción
Colaborador.dni	CHAR(8)	X	X	DNI del consejero, que es un tipo de colaborador.
Sede.id	INT		X	ID de la sede donde trabaja el consejero.

Table 9: Director

Nombre del campo	Tipo de dato	PK	FK	Descripción
id	INT	X		ID del grado.
nombre	VARCHAR(50)			Nombre del grado.

Table 10: Grado

Nombre del campo	Tipo de dato	PK	FK	Descripción
ruc	CHAR(11)	X		RUC de la institución educativa.
descripcion	VARCHAR(1000)			Descripción de la institución.
fundador	VARCHAR(100)			Nombre del fundador de la institución.
fundacionFecha	DATE			Fecha de fundación de la institución.
bannerUrl	VARCHAR(255)			URL del banner de la institución.
nombre	VARCHAR(150)			Nombre de la institución educativa.

Table 11: Institucion

Nombre del campo	Tipo de dato	PK	FK	Descripción
Alumno.dni	CHAR(8)	X	X	DNI del alumno matriculado.
year	INT	X		Año de la matrícula.
Sede.id	INT	X	X	ID de la sede donde el alumno está matriculado.
Grado.id	INT		X	Grado en el que el alumno está matriculado.
Secretario.dni	CHAR(8)		X	DNI del secretario que realizó la matrícula.

Table 12: Matricula

Nombre del campo	Tipo de dato	PK	FK	Descripción
dni	CHAR(8)	X		DNI de la persona.
nombres	VARCHAR(100)			Nombres completos de la persona.
primerApellido	VARCHAR(50)			Primer apellido de la persona.
segundoApellido	VARCHAR(50)			Segundo apellido de la persona.
nacimientoFecha	DATE			Fecha de nacimiento de la persona.
sexo	CHAR(1)			Sexo de la persona.
email	VARCHAR(100)			Email de la persona.

Table 13: Persona

Nombre del campo	Tipo de dato	PK	FK	Descripción
Profesor.dni	CHAR(8)	X	X	DNI del profesor que imparte el curso.
Curso.id	INT	X	X	ID del curso que se imparte.
Grado.id	INT	X	X	ID del grado para el que se imparte el curso.
periodoAcademico	INT			Periodo académico en el que dicta el profesor.

Table 14: ProfesorCursoGrado

Nombre del campo	Tipo de dato	PK	FK	Descripción
Profesor.dni	CHAR(8)	X	X	DNI del profesor.
Sede.id	INT		X	Sede en la que trabaja el profesor.

Table 15: ProfesorSede

Nombre del campo	Tipo de dato	PK	FK	Descripción
Colaborador.dni	CHAR(8)	X	X	DNI del profesor, que es un tipo de colaborador.

Table 16: Profesor

Nombre del campo	Tipo de dato	PK	FK	Descripción
aforo	INT			Capacidad máxima de estudiantes en el salón.
Grado.id	INT		X	ID del grado al que pertenece el salón.
nombreDeSeccion	VARCHAR(50)	X		Nombre de la sección del salón.
Tutor.dni	CHAR(8)		X	DNI del tutor asignado al salón.
Sede.id	INT	X	X	ID de la sede a la que pertenece el salón.

Table 17: Salon

Nombre del campo	Tipo de dato	PK	FK	Descripción
Colaborador.dni	CHAR(8)	X	X	DNI del secretario, que es un tipo de colaborador.
Sede.id	INT		X	ID de la sede donde trabaja el secretario.

Table 18: Secretario

Nombre del campo	Tipo de dato	PK	FK	Descripción
id	INT	X		ID de la sede.
coordenadaLongitud	DOUBLE			Longitud geográfica de la sede.
coordenadaLatitud	DOUBLE			Latitud geográfica de la sede.
direccion	VARCHAR(255)			Dirección física de la sede.
construccionFecha	DATETIME			Fecha de construcción de la sede.
Institucion.ruc	CHAR(11)		X	RUC de la institucion.

Table 19: Sede

Nombre del campo	Tipo de dato	PK	FK	Descripción
Colaborador.dni	CHAR(8)	X	X	DNI del tutor, que es un tipo de colaborador.
nombreDeSeccion	VARCHAR(50)	X		Nombre de la sección del salón.
Sede.id	INT		X	ID de la sede donde trabaja el tutor.

Table 20: Tutor

## 4 Implementación de la base de datos

### 4.1 Creación de tablas en PostgreSQL

```
1 CREATE TABLE institucion
2 (
3     ruc            CHAR(11) PRIMARY KEY,
4     descripcion    VARCHAR(1000) NOT NULL,
5     fundador       VARCHAR(100)  NOT NULL,
6     fundacion_fecha DATE          NOT NULL,
7     banner_url     VARCHAR(255)   NOT NULL,
8     nombre         VARCHAR(150) UNIQUE NOT NULL,
9     CHECK (ruc NOT LIKE '%[~0-9]%'),
10    CHECK (banner_url LIKE 'https://%'),
11    CHECK (fundacion_fecha <= CURRENT_DATE)
12 );
13
14 CREATE TABLE persona
15 (
16     dni            CHAR(8) PRIMARY KEY,
17     nombres        VARCHAR(100)  NOT NULL,
18     primer_apellido VARCHAR(50)   NOT NULL,
19     segundo_apellido VARCHAR(50)  NOT NULL,
20     nacimiento_fecha DATE         NOT NULL,
21     sexo           CHAR(1)        NOT NULL,
22     email          VARCHAR(100) UNIQUE NOT NULL,
23     CHECK (dni NOT LIKE '%[~0-9]%'),
24     CHECK (sexo IN ('M', 'F')),
25     CHECK (email LIKE '%_@_%._%'),
26     CHECK (nacimiento_fecha <= CURRENT_DATE)
27 );
28
29 CREATE TABLE colaborador
30 (
31     dni            CHAR(8) PRIMARY KEY REFERENCES persona
32     (dni),
33     sueldo_hora    FLOAT          NOT NULL,
34     cci            CHAR(20)       NOT NULL,
35     numero_celular VARCHAR(15)   NOT NULL,
36     horas_semanales_trabajo INT    NOT NULL,
37     esta_activo    BOOLEAN        NOT NULL,
38     CHECK (sueldo_hora > 0.0),
39     CHECK (cci NOT LIKE '%[~0-9]%'),
40     CHECK (numero_celular LIKE '+%[0-9 ]%' OR numero_celular NOT
41     LIKE '%[~0-9 ]%'),
42     CHECK (horas_semanales_trabajo BETWEEN 1 AND 60)
43 );
44
45 CREATE TABLE sede
46 (
47     id            SERIAL PRIMARY KEY,
48     coordenada_longitud DOUBLE PRECISION NOT NULL,
49     coordenada_latitud DOUBLE PRECISION NOT NULL,
50     direccion     VARCHAR(255)   NOT NULL,
51     construccion_fecha DATE       NOT NULL,
52     institucion_ruc CHAR(11) REFERENCES institucion (ruc) NOT
53     NULL,
```



```

51     CHECK (coordenada_longitud BETWEEN -180 AND 180),
52     CHECK (coordenada_latitud BETWEEN -90 AND 90),
53     CHECK (construccion_fecha <= CURRENT_DATE)
54 );
55
56 CREATE TABLE director
57 (
58     dni          CHAR(8) PRIMARY KEY REFERENCES colaborador (dni),
59     sede_id INT REFERENCES sede (id) UNIQUE NOT NULL
60 );
61
62 CREATE TABLE consejero
63 (
64     dni          CHAR(8) PRIMARY KEY REFERENCES colaborador (dni),
65     sede_id INT REFERENCES sede (id) NOT NULL
66 );
67
68 CREATE TABLE secretario
69 (
70     dni          CHAR(8) PRIMARY KEY REFERENCES colaborador (dni),
71     sede_id INT REFERENCES sede (id) NOT NULL
72 );
73
74 CREATE TABLE profesor
75 (
76     dni CHAR(8) PRIMARY KEY REFERENCES colaborador (dni)
77 );
78
79 CREATE TABLE grado
80 (
81     id          SERIAL PRIMARY KEY,
82     nombre VARCHAR(50) UNIQUE NOT NULL
83 );
84
85 CREATE TABLE curso
86 (
87     id          SERIAL PRIMARY KEY,
88     nombre VARCHAR(50) NOT NULL
89 );
90
91 CREATE TABLE salon
92 (
93     aforo          INT NOT NULL,
94     nombre_seccion VARCHAR(50) NOT NULL,
95     grado_id       INT REFERENCES grado (id) NOT NULL,
96     sede_id        INT REFERENCES sede (id) NOT NULL,
97     PRIMARY KEY (nombre_seccion, sede_id),
98     CHECK (aforo >= 5 AND aforo <= 40)
99 );
100
101 CREATE TABLE tutor
102 (
103     dni          CHAR(8) PRIMARY KEY REFERENCES colaborador
104     (dni),
105     salon_nombre_seccion VARCHAR(50) NOT NULL,
106     sede_id       INT NOT NULL,
107     FOREIGN KEY (salon_nombre_seccion, sede_id) REFERENCES salon (

```

```

107     nombre_seccion, sede_id),
108     UNIQUE (salon_nombre_seccion, sede_id)
109 );
110 CREATE TABLE apoderado
111 (
112     dni            CHAR(8) PRIMARY KEY REFERENCES persona (dni),
113     numero_celular VARCHAR(15) NOT NULL,
114     CHECK (numero_celular LIKE '+%[0-9 ]%' OR numero_celular NOT
115     LIKE '%[-0-9 ]%')
116 );
117 CREATE TABLE alumno
118 (
119     dni            CHAR(8) PRIMARY KEY REFERENCES persona (
120     dni),
121     salon_nombre_seccion VARCHAR(50) NOT
122     NULL,
123     salon_sede_id    INT NOT
124     NULL,
125     apoderado_dni    CHAR(8) REFERENCES apoderado (dni) NOT
126     NULL,
127     FOREIGN KEY (salon_nombre_seccion, salon_sede_id) REFERENCES
128     salon (nombre_seccion, sede_id)
129 );
130 CREATE TABLE profesor_sede
131 (
132     profesor_dni CHAR(8) REFERENCES profesor (dni),
133     sede_id      INT REFERENCES sede (id) NOT NULL,
134     PRIMARY KEY (profesor_dni, sede_id)
135 );
136 CREATE TABLE profesor_curso_grado
137 (
138     curso_id        INT REFERENCES curso (id),
139     grado_id        INT REFERENCES grado (id),
140     profesor_dni    CHAR(8) REFERENCES profesor (dni),
141     periodo_academico INT NOT NULL,
142     PRIMARY KEY (curso_id, grado_id, profesor_dni),
143     CHECK (periodo_academico <= EXTRACT(YEAR FROM CURRENT_DATE))
144 );
145 CREATE TABLE matricula
146 (
147     year            INT NOT NULL,
148     alumno_dni      CHAR(8) REFERENCES alumno (dni),
149     sede_id         INT REFERENCES sede (id),
150     grado_id        INT REFERENCES grado (id) NOT NULL,
151     secretario_dni  CHAR(8) REFERENCES secretario (dni) NOT NULL,
152     PRIMARY KEY (year, alumno_dni, sede_id),
153     CHECK (year <= EXTRACT(YEAR FROM CURRENT_DATE))
154 );

```

## 4.2 Creación de triggers en PostgreSQL

```

1 CREATE OR REPLACE FUNCTION

```

```

function_check_alumno_overlapping_colaborador_apoderado()
RETURNS TRIGGER AS
2 $$
3 BEGIN
4     IF EXISTS (SELECT 1 FROM colaborador WHERE dni = new.dni) OR
5        EXISTS (SELECT 1 FROM apoderado WHERE dni = new.dni) THEN
6         RAISE EXCEPTION 'La persona con DNI % ya existe en las
7        tablas Colaborador o Apoderado.', new.dni;
8     END IF;
9
10    RETURN new;
11 END;
12 $$ LANGUAGE plpgsql;
13 CREATE TRIGGER trigger_check_alumno_overlapping
14     BEFORE INSERT
15     ON alumno
16     FOR EACH ROW
17 EXECUTE FUNCTION
18     function_check_alumno_overlapping_colaborador_apoderado();
19 CREATE OR REPLACE FUNCTION
20     function_check_colaborador_sueldo_mensual() RETURNS TRIGGER AS
21 $$
22 DECLARE
23     sueldo_mensual INT;
24 BEGIN
25     sueldo_mensual = new.sueldo_hora * new.horas_semanales_trabajo
26     * 4;
27
28     IF sueldo_mensual < 1025 THEN
29         RAISE EXCEPTION 'El sueldo mensual debe ser % mayor al
30         sueldo minimo de 1025.', sueldo_mensual;
31     END IF;
32
33     RETURN new;
34 END;
35 $$ LANGUAGE plpgsql;
36 CREATE TRIGGER trigger_check_colaborador_sueldo_mensual
37     BEFORE INSERT OR UPDATE
38     ON colaborador
39     FOR EACH ROW
40 EXECUTE FUNCTION function_check_colaborador_sueldo_mensual();
41
42 ----
43
44 CREATE OR REPLACE FUNCTION function_check_colaborador_overlapping()
45     RETURNS TRIGGER AS
46 $$
47 BEGIN
48     IF (EXISTS (SELECT 1 FROM profesor WHERE dni = new.dni)) OR
49        (EXISTS (SELECT 1 FROM consejero WHERE dni = new.dni)) OR
50        (EXISTS (SELECT 1 FROM secretario WHERE dni = new.dni)) OR
51        (EXISTS (SELECT 1 FROM director WHERE dni = new.dni)) OR
52        (EXISTS (SELECT 1 FROM tutor WHERE dni = new.dni)) THEN
53         RAISE EXCEPTION 'El colaborador con DNI % ya existe en otra

```

```

        tabla_hija.', new.dni;
51     END IF;
52
53     RETURN new;
54 END;
55 $$ LANGUAGE plpgsql;
56
57 CREATE TRIGGER trigger_check_profesor_overlapping
58     BEFORE INSERT
59     ON profesor
60     FOR EACH ROW
61 EXECUTE FUNCTION function_check_colaborador_overlapping();
62
63 CREATE TRIGGER trigger_check_consejero_overlapping
64     BEFORE INSERT
65     ON consejero
66     FOR EACH ROW
67 EXECUTE FUNCTION function_check_colaborador_overlapping();
68
69 CREATE TRIGGER trigger_check_secretario_overlapping
70     BEFORE INSERT
71     ON secretario
72     FOR EACH ROW
73 EXECUTE FUNCTION function_check_colaborador_overlapping();
74
75 CREATE TRIGGER trigger_check_director_overlapping
76     BEFORE INSERT
77     ON director
78     FOR EACH ROW
79 EXECUTE FUNCTION function_check_colaborador_overlapping();
80
81 CREATE TRIGGER trigger_check_tutor_overlapping
82     BEFORE INSERT
83     ON tutor
84     FOR EACH ROW
85 EXECUTE FUNCTION function_check_colaborador_overlapping();
86
87 ----
88
89 CREATE OR REPLACE FUNCTION
90     function_check_colaborador_apoderado_overlapping_alumno()
91     RETURNS TRIGGER AS
92 $$
93 BEGIN
94     IF EXISTS (SELECT 1 FROM alumno WHERE dni = new.dni) THEN
95         RAISE EXCEPTION 'La persona con DNI % ya existe en la tabla
96             Alumno.', new.dni;
97     END IF;
98
99     RETURN new;
100 END;
101 $$ LANGUAGE plpgsql;
102
103 CREATE TRIGGER trigger_check_colaborador_overlapping
104     BEFORE INSERT
105     ON colaborador
106     FOR EACH ROW

```

```

104 EXECUTE FUNCTION
      function_check_colaborador_apoderado_overlapping_alumno();
105
106 CREATE TRIGGER trigger_check_apoderado_overlapping
107     BEFORE INSERT
108     ON apoderado
109     FOR EACH ROW
110 EXECUTE FUNCTION
      function_check_colaborador_apoderado_overlapping_alumno();
111
112 ----
113
114 CREATE OR REPLACE FUNCTION function_check_colaborador_esta_activo()
      RETURNS TRIGGER AS
115 $$
116 DECLARE
117     esta_activo BOOLEAN;
118 BEGIN
119     SELECT esta_activo
120     INTO esta_activo
121     FROM colaborador
122     WHERE new.dni = dni;
123
124     IF NOT esta_activo THEN
125         RAISE EXCEPTION 'El colaborador que se intenta insertar no
esta activo.';
126     END IF;
127
128     RETURN new;
129 END;
130 $$ LANGUAGE plpgsql;
131
132 CREATE TRIGGER trigger_check_profesor_esta_activo
133     BEFORE INSERT OR UPDATE
134     ON persona
135     FOR EACH ROW
136 EXECUTE FUNCTION function_check_colaborador_esta_activo();
137
138 ----
139
140 CREATE OR REPLACE FUNCTION gestionar_director_reasignacion()
      RETURNS TRIGGER AS
141 $$
142 BEGIN
143     IF (SELECT COUNT(*) FROM director WHERE sede_id = old.sede_id
AND dni != old.dni) = 0 THEN
144         RAISE EXCEPTION 'No se puede reasignar el director sin
reemplazo en la sede %', old.sedeid;
145     END IF;
146
147     DELETE FROM director WHERE dni = old.dni;
148
149     UPDATE colaborador SET esta_activo = FALSE WHERE dni = old.dni;
150
151     RETURN new;
152 END;
153 $$ LANGUAGE plpgsql;

```

```

154
155 CREATE TRIGGER trigger_gestionar_director_reasignacion
156     AFTER UPDATE
157     ON director
158     FOR EACH ROW
159     WHEN (old.dni IS DISTINCT FROM new.dni)
160 EXECUTE FUNCTION gestionar_director_reasignacion();
161
162 ----
163
164 CREATE OR REPLACE FUNCTION function_check_matricula_year() RETURNS
    TRIGGER AS
165 $$
166 BEGIN
167     IF new.year <= (SELECT EXTRACT(YEAR FROM construccion_fecha)
168 FROM sede WHERE id = new.sede_id) THEN
169         RAISE EXCEPTION 'El anho de matricula debe ser mayor que el
170 anho de construccion de la sede.';
171     END IF;
172     RETURN new;
173 END;
174 $$ LANGUAGE plpgsql;
175
176 CREATE TRIGGER trigger_check_matricula_year
177     BEFORE INSERT OR UPDATE
178     ON matricula
179     FOR EACH ROW
180 EXECUTE FUNCTION function_check_matricula_year();
181
182 ----
183
184 CREATE OR REPLACE FUNCTION function_check_salon_aforo_on_alumno()
185     RETURNS TRIGGER AS
186 $$
187 DECLARE
188     salon_aforo          INT;
189     alumnos_cantidad INT;
190 BEGIN
191     SELECT aforo
192     INTO salon_aforo
193     FROM salon
194     WHERE nombre_seccion = new.nombre_seccion
195           AND sede_id = new.sede_id;
196
197     SELECT COUNT(dni) + 1
198     INTO alumnos_cantidad
199     FROM alumno
200     WHERE salon_nombre_seccion = new.salon_nombre_seccion
201           AND salon_sede_id = new.salon_sede_id;
202
203     IF alumnos_cantidad > salon_aforo THEN
204         RAISE EXCEPTION 'El aforo del salon ha sido excedido. Aforo
205 maximo: %, Numero de alumnos: %', salon_aforo,
206 alumnos_cantidad;
207     END IF;
208
209     RETURN new;
210

```

```

206 END;
207 $$ LANGUAGE plpgsql;
208
209 CREATE TRIGGER trigger_check_salon_aforo_on_alumno
210 BEFORE INSERT
211 ON alumno
212 FOR EACH ROW
213 EXECUTE FUNCTION function_check_salon_aforo_on_alumno();
214
215 ----
216
217 CREATE OR REPLACE FUNCTION function_check_salon_aforo_on_matricula
218 ()
219 RETURNS TRIGGER AS
220 $$
221 DECLARE
222     salon_aforo          INT;
223     alumnos_cantidad     INT;
224     salon_nombre_seccion VARCHAR(50);
225     salon_sede_id        INT;
226 BEGIN
227     SELECT salon_nombre_seccion, salon_sede_id
228     INTO salon_nombre_seccion, salon_sede_id
229     FROM alumno
230     WHERE dni = new.alumno_dni;
231
232     SELECT aforo
233     INTO salon_aforo
234     FROM salon
235     WHERE salon_nombre_seccion = new.nombre_seccion
236           AND salon_sede_id = new.sede_id;
237
238     SELECT COUNT(dni) + 1
239     INTO alumnos_cantidad
240     FROM alumno
241     WHERE salon_nombre_seccion = new.nombre_seccion
242           AND salon_sede_id = new.sede_id;
243
244     IF alumnos_cantidad > salon_aforo THEN
245         RAISE EXCEPTION 'El aforo del salon ha sido excedido. Aforo
246             maximo: %, Numero de alumnos: %', salon_aforo,
247             alumnos_cantidad + 1;
248     END IF;
249
250     RETURN new;
251 END;
252 $$ LANGUAGE plpgsql;
253
254 CREATE TRIGGER trigger_check_salon_aforo_on_matricula
255 BEFORE INSERT
256 ON matricula
257 FOR EACH ROW
258 EXECUTE FUNCTION function_check_salon_aforo_on_matricula();
259
260 ----
261
262 CREATE OR REPLACE FUNCTION gestionar_tutor_reasignacion()

```

```

260 RETURNS TRIGGER AS
261 $$
262 BEGIN
263     IF (SELECT COUNT(*)
264         FROM tutor
265         WHERE salon_nombre_seccion = old.salon_nombre_seccion
266             AND sede_id = old.sede_id
267             AND dni != old.dni) = 0 THEN
268         RAISE EXCEPTION 'No se puede reasignar el tutor sin
reemplazo en el salon % de la sede %', old.salon_nombre_seccion
, old.sede_id;
269     END IF;
270
271     DELETE FROM tutor WHERE dni = old.dni AND salon_nombre_seccion
= old.salon_nombre_seccion AND sede_id = old.sede_id;
272
273     UPDATE colaborador SET esta_activo = FALSE WHERE dni = old.dni;
274
275     RETURN new;
276 END;
277 $$ LANGUAGE plpgsql;
278
279 CREATE TRIGGER trigger_gestionar_tutor_reasignacion
280 AFTER UPDATE
281 ON tutor
282 FOR EACH ROW
283 WHEN (old.dni IS DISTINCT FROM new.dni OR old.
salon_nombre_seccion IS DISTINCT FROM new.salon_nombre_seccion
OR
284         old.sede_id IS DISTINCT FROM new.sede_id)
285 EXECUTE FUNCTION gestionar_tutor_reasignacion();

```

### 4.3 Carga de datos

Durante la carga de datos en los esquemas de 1k, 10k, 100k y 1m, se implementó la simulación de datos faltantes en archivos CSV, seguida de su inserción mediante Docker como tuplas en lugar de listas. Este enfoque fue diseñado para optimizar las operaciones de lectura desde los contenedores. El propósito principal de esta práctica es facilitar el acceso inmediato a la base de datos simplemente descargando la imagen Docker, asegurando así una configuración ágil y eficiente del entorno de desarrollo.

### 4.4 Simulación de datos faltantes

Para simular datos faltantes, se desarrolló un script en Python que utilizó la biblioteca externa Faker junto con la biblioteca nativa random para generar datos aleatorios en los archivos CSV mencionados. Estos datos fueron posteriormente insertados mediante un bulk insert, optimizando así el proceso de carga masiva y garantizando la diversidad y precisión de los datos simulados.



## 5 Optimización y experimentación

En la siguiente sección, se evaluará el rendimiento de la base de datos mediante la ejecución de tres consultas complejas. Este análisis se llevará a cabo en varios escenarios que incluyen diferentes volúmenes de datos, y se realizarán pruebas sin índices, únicamente con los índices por defecto y con los índices que consideremos más adecuados para garantizar la ejecución óptima de estas consultas. Finalmente, se procederá a analizar y comparar los resultados obtenidos.

### 5.1 Consultas SQL para el experimento

#### 5.1.1 Descripción del tipo de consultas seleccionadas

- **Consulta 1:** La siguiente consulta obtiene el nombre, email y número de celular de los directores actuales, así como la dirección, coordenadas y fecha de construcción de las 20 sedes más antiguas construidas entre 1990 y 2010, excluyendo aquellas sedes donde el número total de profesores y alumnos exceda 400.
  - **Justificación:** La institución educativa está en un proceso de planificación para realizar renovaciones y mantenimiento en sus sedes. Se ha decidido comenzar con las sedes construidas entre los años 1990 y 2010, ya que estas son las que han demostrado tener más necesidad de atención. Para minimizar las interrupciones en las actividades escolares, se ha establecido que solo se seleccionarán aquellas sedes donde la suma del número de profesores y alumnos no exceda 400. Además, se requiere la información de los directores de estas sedes para coordinar las visitas de evaluación y supervisión.
- **Consulta 2:** Se desea conocer el nombre completo, bonificación, código de cuenta interbancaria e email de los colaboradores identificados como activos actualmente a los que les corresponde el bono que ofrece la institución. Para calcular la bonificación debemos tomar en cuenta que las sedes que este año están cumpliendo un aniversario múltiplo de 10 (sin contar al 0) ofrecen un bono de 5% respecto al pago mensual (el sueldo mensual se obtiene cuadruplicando la multiplicación del pago por hora por las horas semanales) para sus colaboradores que nacieron entre 1960 y 1980.
  - **Justificación:** La institución tiene una política que consiste en que cada 10 años desde la construcción de cada sede se ofrece un bono de 5% respecto al pago mensual, a los colaboradores activos mayores que trabajan en esa sede. Para obtener el monto de bonificación se debe cuadruplicar la multiplicación de la remuneración establecida por hora por la cantidad de horas semanales laboradas. De existir un colaborador que labore en más de una sede se debe evaluar la condición del aniversario para todas las sedes; si se cumple la condición en

más de una, debemos multiplicar el bono por el número de sedes en las que se cumpla. El área de finanzas debe realizar el desembolso de este bono; por ello, se necesita conocer el nombre completo, monto respectivo, código de cuenta interbancaria para realizar la transferencia de este, y su email para enviar el comprobante de pago.

- **Consulta 3:** Se desea conocer el nombre, primer apellido, segundo apellido, sexo e email de los alumnos menores de 18 años cuyo apoderado sea un colaborador registrado como activo, que trabaja a tiempo completo (más de 48 horas semanales) y cuyo sueldo mensual sea menor a 2000 soles (el sueldo mensual se obtiene cuadruplicando la multiplicación del pago por hora por las horas semanales). Además deben haber transcurrido como mínimo 2 años desde la matrícula del alumno.
  - **Justificación:** La institución educativa va a implementar la iniciativa de ofrecer un descuento especial en el pago mensual a los alumnos menores de 18 años cuyo apoderado sea un colaborador de la institución, siempre y cuando esté registrado como activo, labore a tiempo completo, su sueldo mensual (resultado de cuadruplicar la multiplicación de sueldo por hora, por las horas semanales) sea menor a 2000. El alumno debe haber estado matriculado por lo menos 2 años antes del actual.

### 5.1.2 Implementación de consultas en SQL

- **Consulta 1:**

```
1 SELECT
2     d_persona.nombres || ' ' || d_persona.primer_apellido || ' '
3     ' ' || d_persona.segundo_apellido AS nombre_director,
4     d_persona.email AS email_director,
5     d_colaborador.numero_celular AS celular_director,
6     sede.direccion,
7     sede.coordenada_longitud,
8     sede.coordenada_latitud,
9     sede.construccion_fecha
10 FROM
11     director
12     JOIN colaborador AS d_colaborador ON director.dni =
13     d_colaborador.dni
14     JOIN persona AS d_persona ON d_colaborador.dni =
15     d_persona.dni
16     JOIN sede ON director.sede_id = sede.id
17 WHERE
18     sede.construccion_fecha BETWEEN '1990-01-01' AND '
19     2010-12-31'
20 AND (
21     SELECT COUNT(*)
22     FROM profesor_sede
23     WHERE profesor_sede.sede_id = sede.id
24 ) + (
25     SELECT COUNT(*)
26     FROM alumno
```

```

23         WHERE alumno.salon_sede_id = sede.id
24     ) <= 400
25 ORDER BY
26     sede.construccion_fecha
27 LIMIT 20;

```

• Consulta 2:

```

1 SELECT CONCAT(persona.nombres, ' ', persona.primer_apellido, '
2     ', persona.segundo_apellido) AS nombre_completo,
3     colaborador.cci,
4     persona.email,
5     CASE
6         WHEN profesor.dni IS NOT NULL THEN
7             colaborador.sueldo_hora * colaborador.
8             horas_semanales_trabajo * 4 * 0.05 *
9             (SELECT COUNT(id)
10              FROM sede
11              JOIN profesor_sede ON sede.id =
12              profesor_sede.sede_id
13              WHERE profesor_sede.profesor_dni = profesor.
14              dni
15              AND (EXTRACT(YEAR FROM CURRENT_DATE) -
16              EXTRACT(YEAR FROM sede.construccion_fecha)) % 10 = 0
17              AND (EXTRACT(YEAR FROM CURRENT_DATE) -
18              EXTRACT(YEAR FROM sede.construccion_fecha)) > 0)
19         ELSE
20             colaborador.sueldo_hora * colaborador.
21             horas_semanales_trabajo * 4 * 0.05 *
22             (CASE
23                 WHEN (EXTRACT(YEAR FROM CURRENT_DATE) -
24                 EXTRACT(YEAR FROM
25                     (SELECT MIN(sede.construccion_fecha)
26                     FROM sede JOIN colaborador AS c ON c.dni =
27                     colaborador.dni
28                     WHERE c.dni = colaborador.dni
29                     AND (EXTRACT(YEAR FROM CURRENT_DATE) -
30                     EXTRACT(YEAR FROM sede.construccion_fecha)) % 10 = 0
31                     AND (EXTRACT(YEAR FROM CURRENT_DATE) -
32                     EXTRACT(YEAR FROM sede.construccion_fecha)) > 0))) >= 10
33                 THEN 1 ELSE 0 END)
34             END AS bonificacion
35 FROM colaborador
36     JOIN persona ON colaborador.dni = persona.dni
37     LEFT JOIN profesor ON colaborador.dni = profesor.dni
38 WHERE colaborador.esta_activo = TRUE
39     AND persona.nacimiento_fecha BETWEEN '1960-01-01' AND '
40     1980-12-31'
41     AND EXISTS (SELECT 1
42                 FROM sede JOIN colaborador AS c ON c.dni =
43                 colaborador.dni
44                 WHERE c.dni = colaborador.dni
45                 AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT
46                 (YEAR FROM sede.construccion_fecha)) % 10 = 0
47                 AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT
48                 (YEAR FROM sede.construccion_fecha)) > 0);

```

- Consulta 3:

```
1 SELECT persona.nombres ,
2     persona.primer_apellido ,
3     persona.segundo_apellido ,
4     persona.sexo ,
5     persona.email
6 FROM persona
7 WHERE persona.dni
8     IN (SELECT alumno.dni
9         FROM alumno
10        WHERE alumno.dni
11            IN (SELECT matricula.alumno_dni
12                FROM matricula
13                WHERE matricula.year <= EXTRACT(YEAR FROM
14                    CURRENT_DATE) - 2)
14            AND alumno.apoderado_dni
15            IN (SELECT apoderado.dni
16                FROM apoderado
17                WHERE apoderado.dni
18                    IN (SELECT colaborador.dni
19                        FROM colaborador
20                        WHERE colaborador.esta_activo
21                            = TRUE
22                            AND colaborador.
23                                horas_semanales_trabajo > 48
24                                AND colaborador.sueldo_hora *
25                                    colaborador.horas_semanales_trabajo * 4 < 2000)))
26 AND persona.nacimiento_fecha > CURRENT_DATE - INTERVAL '18
27 years';
```

## 5.2 Metodología del experimento

Primero, creamos cuatro esquemas: “mil\_datos”, “diezmil\_datos”, “cienmil\_datos” y “millon\_datos”. Cada uno de estos esquemas contiene la cantidad de datos que su nombre indica.

```
1 CREATE SCHEMA mil_datos;
2 CREATE SCHEMA diezmil_datos;
3 CREATE SCHEMA cienmil_datos;
4 CREATE SCHEMA millon_datos;
```

Luego, ejecutaremos cada consulta en cada uno de los esquemas, primero, sin índices, segundo, con los índices por defecto y, finalmente, con los índices por defecto más los índices que consideremos más adecuados para garantizar la ejecución óptima de estas consultas.

Finalmente, mediremos el tiempo de ejecución y analizaremos los planes de ejecución de cada consulta en cada uno de los escenarios mencionados.

## 5.3 Optimización de consultas

### 5.3.1 Planes de índices para la primera consulta

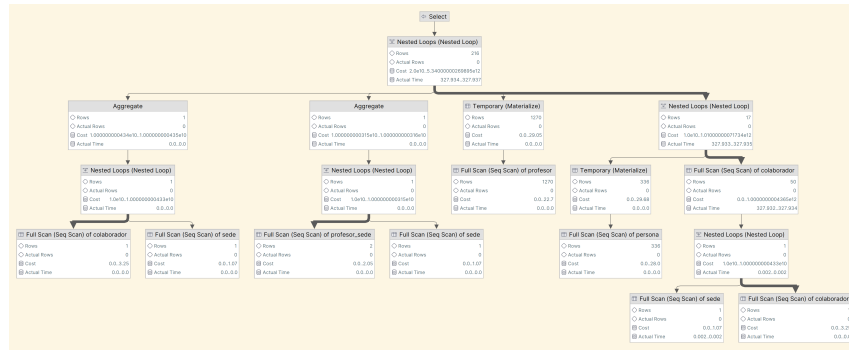
- Ejecución sin índices

```

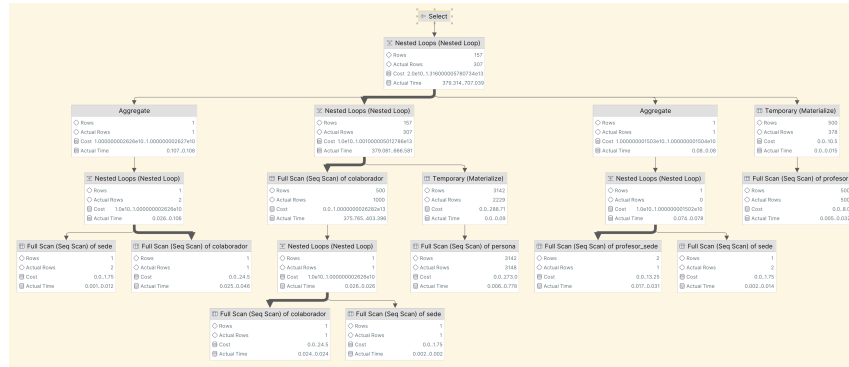
1 SET enable_mergejoin TO OFF;
2 SET enable_hashjoin TO OFF;
3 SET enable_bitmapscan TO OFF;
4 SET enable_sort TO OFF;
5 SET enable_nestloop TO OFF;
6 SET enable_indexscan TO OFF;
7 SET enable_indexonlyscan TO OFF;
8
9 DROP INDEX IF EXISTS idx_persona_nacimiento_fecha;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL persona;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL sede;

```

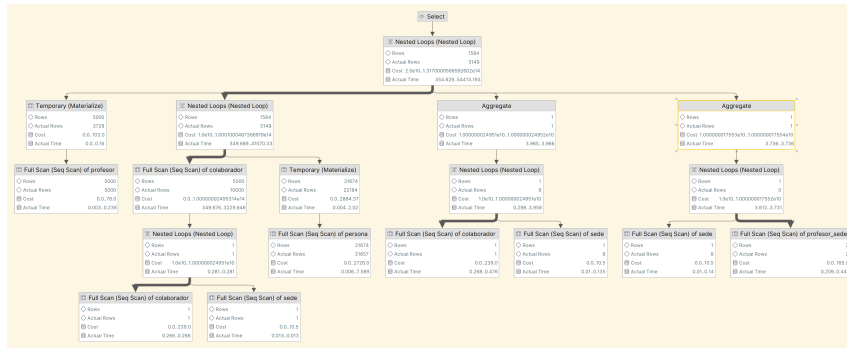
– Para mil datos:



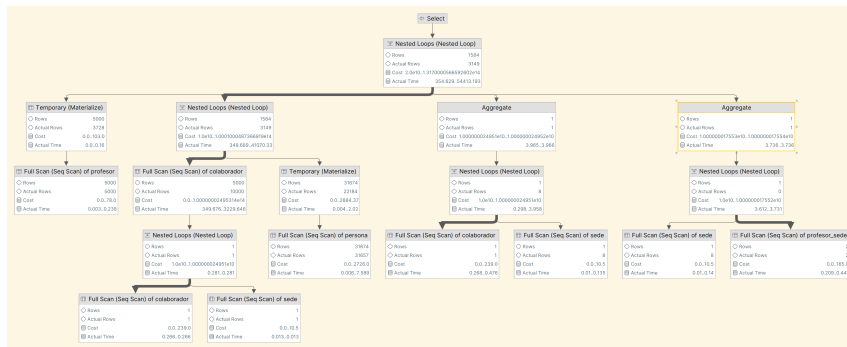
– Para diez mil datos:



– Para cien mil datos:



– Para un millón de datos:



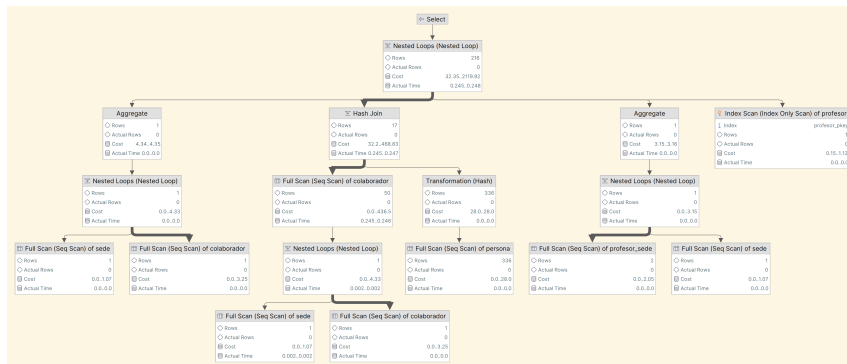
## • Ejercución con índices por defecto

```

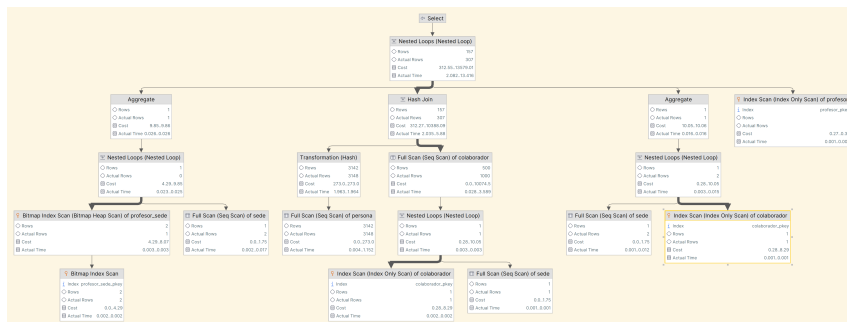
1 SET enable_mergejoin TO ON;
2 SET enable_hashjoin TO ON;
3 SET enable_bitmapscan TO ON;
4 SET enable_sort TO ON;
5 SET enable_nestloop TO ON;
6 SET enable_indexscan TO ON;
7 SET enable_indexonlyscan TO ON;
8
9 DROP INDEX IF EXISTS idx_persona_nacimiento_fecha;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL persona;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL sede;

```

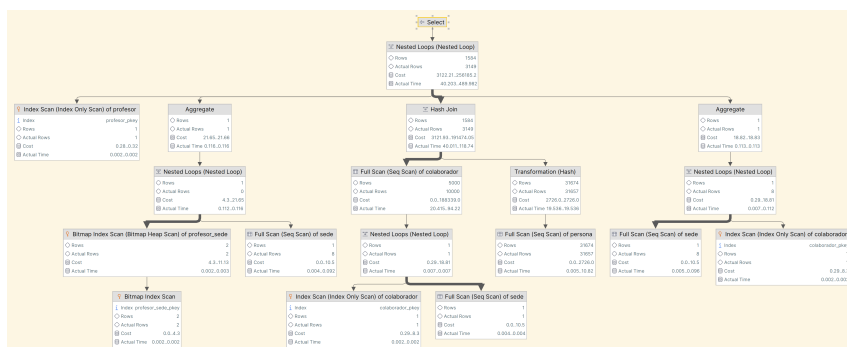
– Para mil datos:

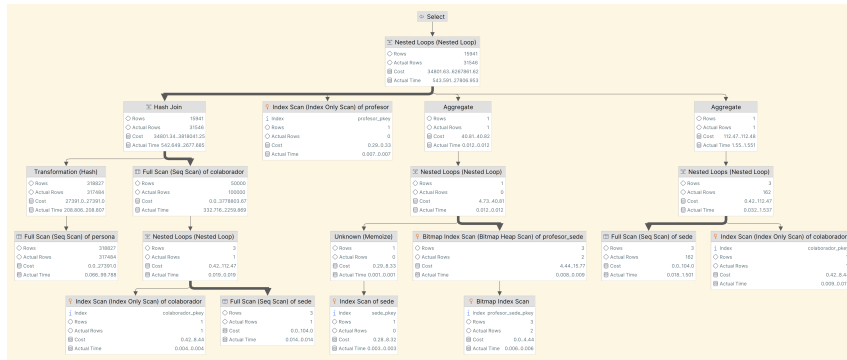


- Para diez mil datos:



- Para cien mil datos:





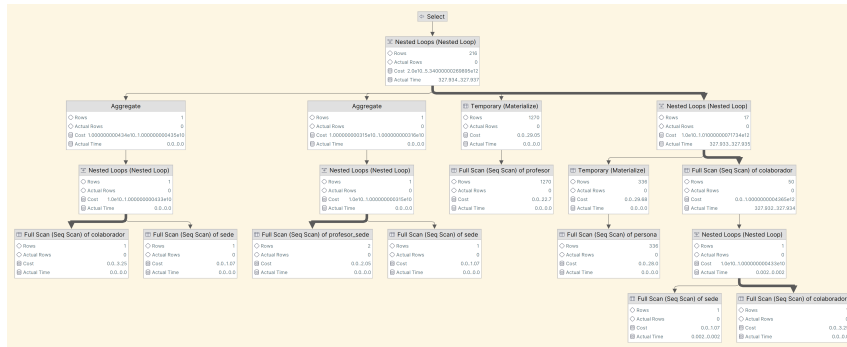
## • Ejecución con índices por defecto más índices personalizados

```

1 SET enable_mergejoin TO OFF;
2 SET enable_hashjoin TO OFF;
3 SET enable_bitmapscan TO OFF;
4 SET enable_sort TO OFF;
5 SET enable_nestloop TO OFF;
6 SET enable_indexscan TO OFF;
7 SET enable_indexonlyscan TO OFF;
8
9 DROP INDEX IF EXISTS idx_persona_nacimiento_fecha;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL persona;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL sede;

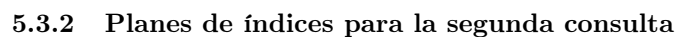
```

– Para mil datos:



– Para diez mil datos:



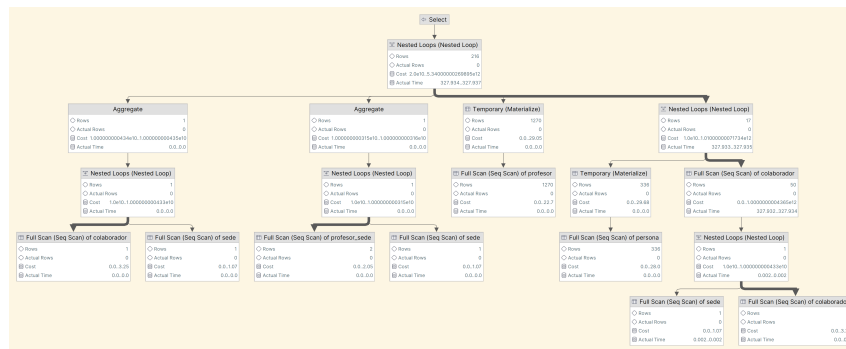


- Ejecución sin índices

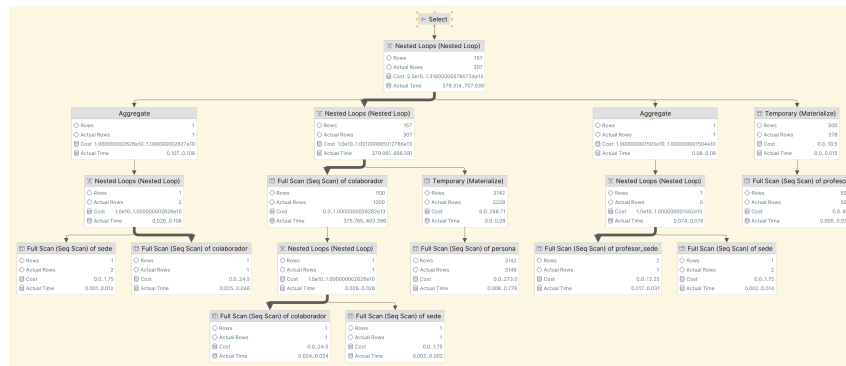
```
1 SET enable_mergejoin TO OFF;
2 SET enable_hashjoin TO OFF;
3 SET enable_bitmapscan TO OFF;
```

10  
11  
12  
13  
14

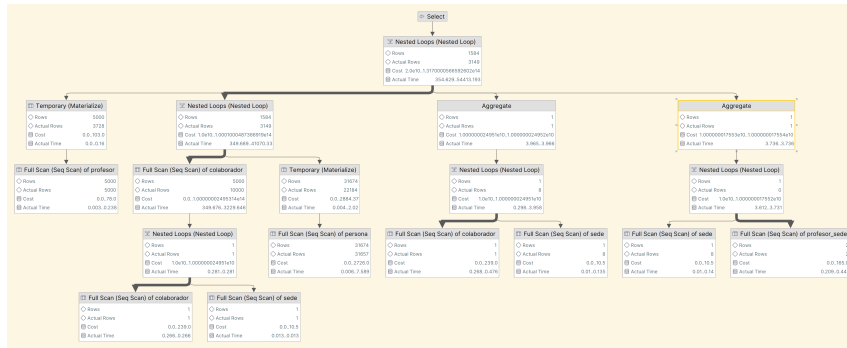
- Para mil datos:



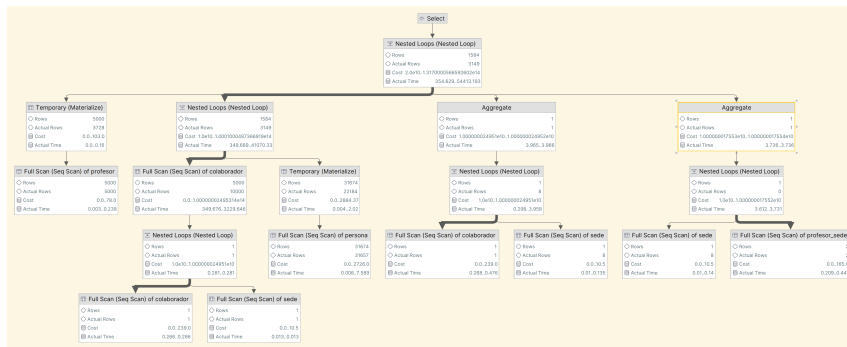
- Para diez mil datos:



- Para cien mil datos:



– Para un millón de datos:



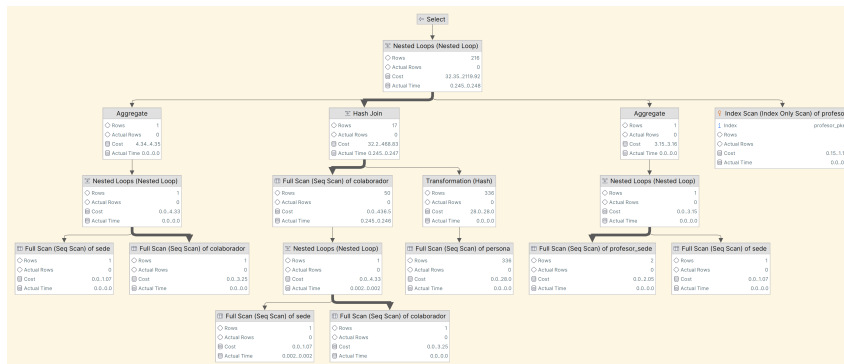
## • Ejercución con índices por defecto

```

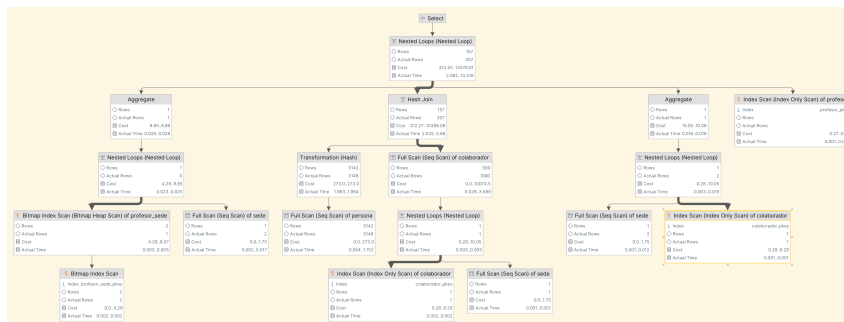
1 SET enable_mergejoin TO ON;
2 SET enable_hashjoin TO ON;
3 SET enable_bitmapscan TO ON;
4 SET enable_sort TO ON;
5 SET enable_nestloop TO ON;
6 SET enable_indexscan TO ON;
7 SET enable_indexonlyscan TO ON;
8
9 DROP INDEX IF EXISTS idx_persona_nacimiento_fecha;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL persona;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL sede;

```

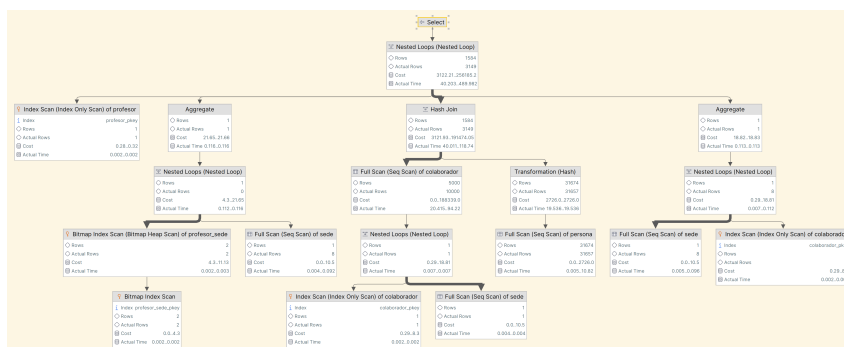
– Para mil datos:

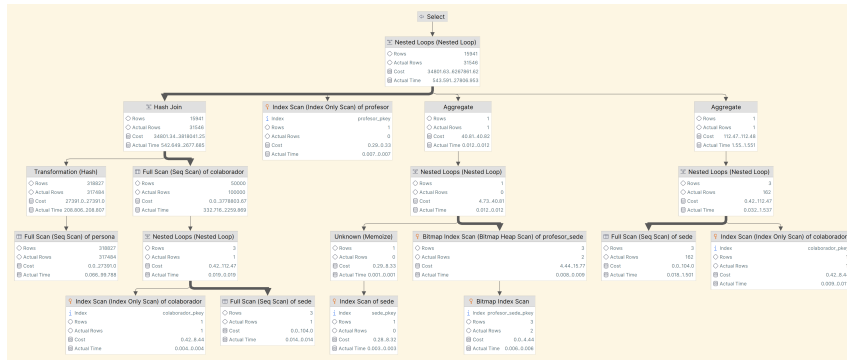


- Para diez mil datos:



- Para cien mil datos:





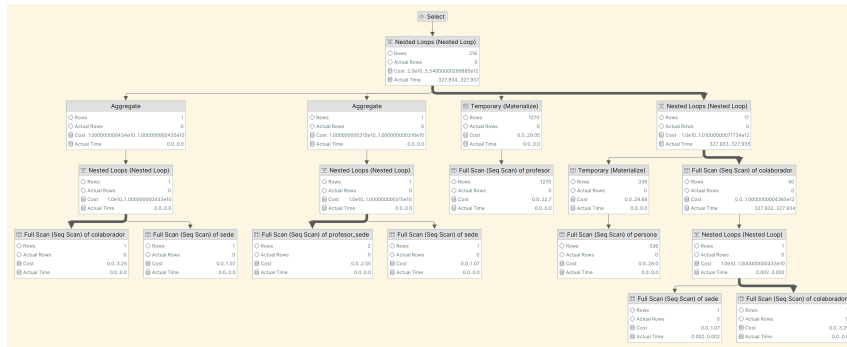
## • Ejecución con índices por defecto más índices personalizados

```

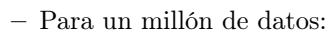
1 SET enable_mergejoin TO OFF;
2 SET enable_hashjoin TO OFF;
3 SET enable_bitmapscan TO OFF;
4 SET enable_sort TO OFF;
5 SET enable_nestloop TO OFF;
6 SET enable_indexscan TO OFF;
7 SET enable_indexonlyscan TO OFF;
8
9 DROP INDEX IF EXISTS idx_persona_nacimiento_fecha;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL persona;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL sede;

```

– Para mil datos:



– Para diez mil datos:



- Ejecución sin índices

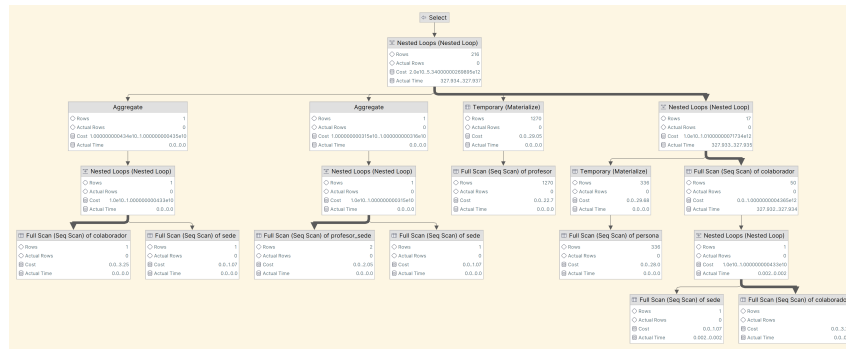
```
1 SET enable_mergejoin TO OFF;
2 SET enable_hashjoin TO OFF;
3 SET enable_bitmapscan TO OFF;
```

```

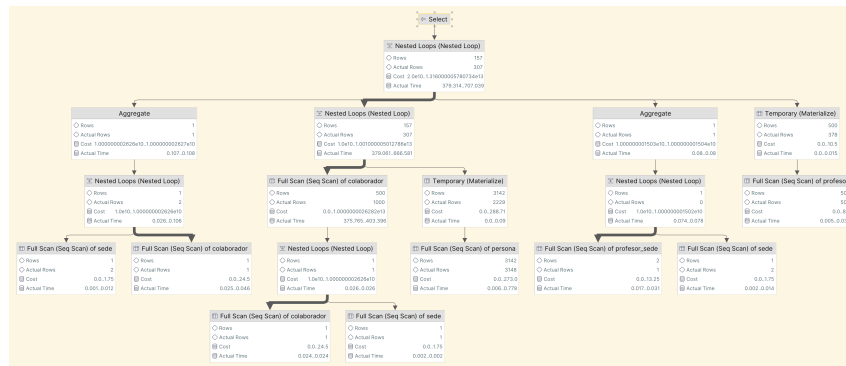
4 SET enable_sort TO OFF;
5 SET enable_nestloop TO OFF;
6 SET enable_indexscan TO OFF;
7 SET enable_indexonlyscan TO OFF;
8
9 DROP INDEX IF EXISTS idx_persona_nacimiento_fecha;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL persona;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL sede;

```

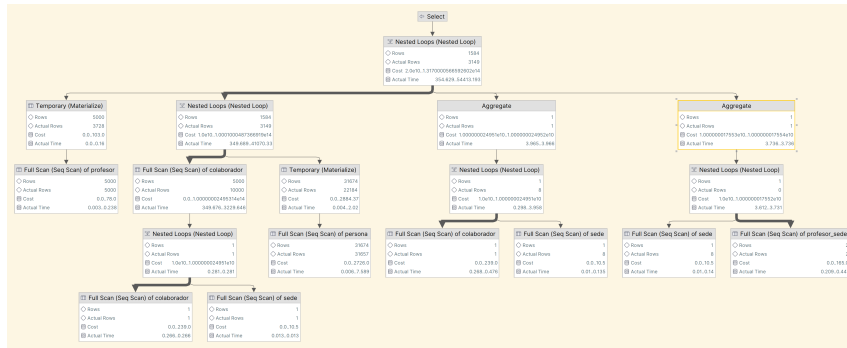
- Para mil datos:



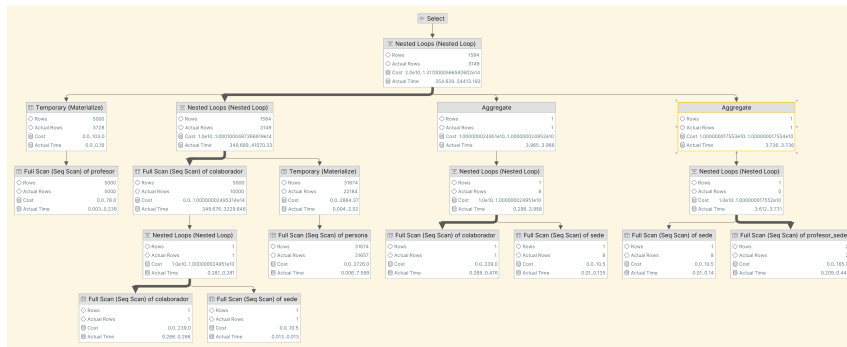
- Para diez mil datos:



- Para cien mil datos:



– Para un millón de datos:



## • Ejercución con índices por defecto

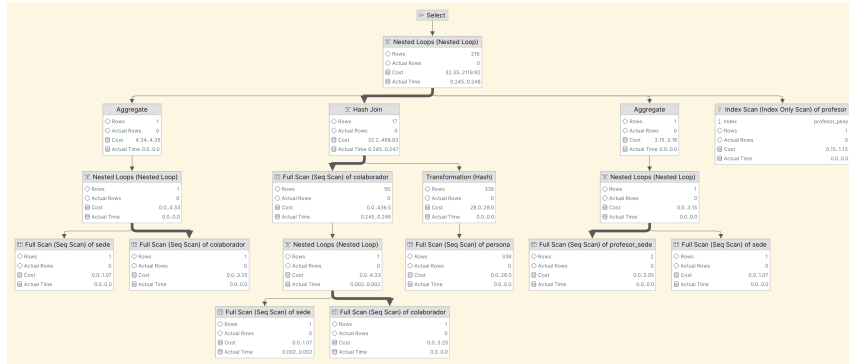
```

1 SET enable_mergejoin TO ON;
2 SET enable_hashjoin TO ON;
3 SET enable_bitmapscan TO ON;
4 SET enable_sort TO ON;
5 SET enable_nestloop TO ON;
6 SET enable_indexscan TO ON;
7 SET enable_indexonlyscan TO ON;
8
9 DROP INDEX IF EXISTS idx_persona_nacimiento_fecha;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL persona;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL sede;

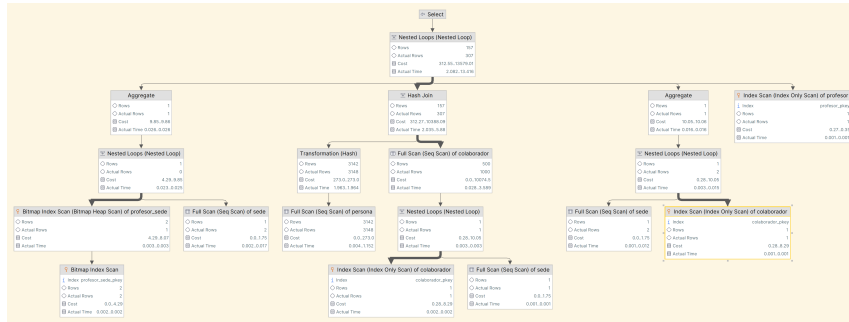
```

– Para mil datos:

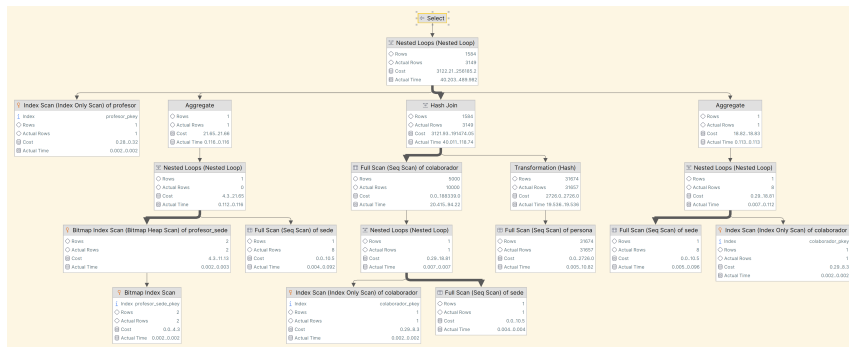




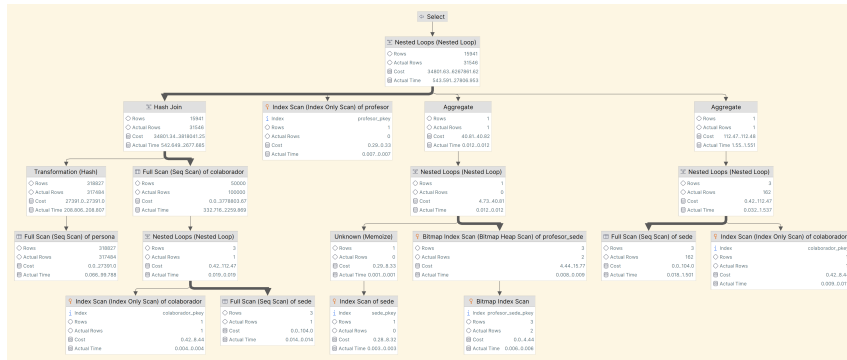
– Para diez mil datos:



– Para cien mil datos:



– Para un millón de datos:



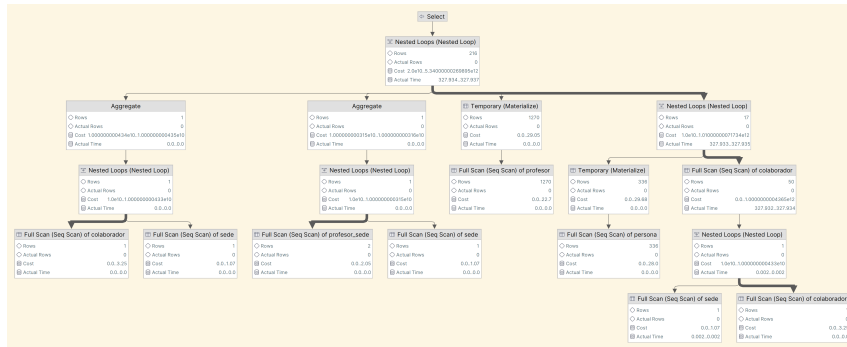
- Ejecución con índices por defecto más índices personalizados

```

1 SET enable_mergejoin TO OFF;
2 SET enable_hashjoin TO OFF;
3 SET enable_bitmapscan TO OFF;
4 SET enable_sort TO OFF;
5 SET enable_nestloop TO OFF;
6 SET enable_indexscan TO OFF;
7 SET enable_indexonlyscan TO OFF;
8
9 DROP INDEX IF EXISTS idx_persona_nacimiento_fecha;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL persona;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL sede;

```

- Para mil datos:



- Para diez mil datos:



## 5.4 Plataforma de pruebas

<b>Sistema Operativo</b>	Windows 10 64-bits
<b>RAM</b>	16 GB
<b>CPU</b>	Intel Core i7-10750H
<b>Capacidad SSD</b>	512 GB
<b>Capacidad HDD</b>	1 TB
<b>PostgreSQL</b>	16.0

Table 21: Especificaciones plataforma de pruebas

## 5.5 Medición de tiempos

### 5.5.1 Sin índices

### 5.5.2 Con índices

## 5.6 Resultados

### 5.6.1 Consulta 1

### 5.6.2 Consulta 2

### 5.6.3 Consulta 3

## 6 Análisis y discusión

## 7 Conclusiones

## 8 Anexos

### 8.1 Modelo Físico

### 8.2 Videos de experimentación

#### 8.2.1 Consulta 1

#### 8.2.2 Consulta 2

#### 8.2.3 Consulta 3

#### 8.2.4 Consulta 4

### 8.3 Pregunta extra

¿Cuál sería la complejidad operacional si escalamos los datos por encima del millón?, realice una comparativa respecto a la cantidad de datos del párrafo anterior. ¿Es suficiente la arquitectura Cliente-Servidor para procesar millones de datos?