

**Universidad de Ingeniería y Tecnología**

**FACULTAD DE COMPUTACIÓN**

**ERP PARA INSTITUCIONES DE  
EDUCACIÓN PRIMARIA Y  
SECUNDARIA**

**HITO 2**

**AUTORES:**

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

**PROFESOR:**

**RIOS OJEDA, BRENNER HUMBERTO**

**JULIO, 2024**

# Contenidos

<b>1 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 . . . . .	3
1.4 ¿Cómo resuelve el problema hoy? . . . . .	3
1.4.1 ¿Cómo se almacenan y procesan los datos? . . . . .	3
1.4.2 Flujo de datos . . . . .	4
1.5 Descripción detallada del sistema . . . . .	4
1.5.1 Objetos de información actuales . . . . .	4
1.5.2 Características y funcionalidades esperadas . . . . .	4
1.5.3 Tipos de usuarios existentes o necesarios . . . . .	4
1.5.4 Tipo de consulta y actualizaciones . . . . .	4
1.5.5 Tamaño estimado de la base de datos . . . . .	5
1.6 Objetivos del proyecto . . . . .	6
1.7 Referencias del proyecto . . . . .	6
1.8 Eventualidades . . . . .	6
1.8.1 Problemas que pudieran encontrarse en el proyecto . . . . .	6
1.8.2 Límites y alcances del proyecto . . . . .	6
<b>2 Modelo Entidad-Relación</b>	<b>7</b>
2.1 Reglas semánticas . . . . .	7
2.2 Modelo Entidad-Relación . . . . .	8
2.3 Especificaciones y consideraciones sobre el modelo . . . . .	9
2.3.1 Entidad Institucion . . . . .	9
2.3.2 Entidad Persona . . . . .	9
2.3.3 Entidad Alumno . . . . .	9
2.3.4 Entidad Apoderado . . . . .	9
2.3.5 Entidad Colaborador . . . . .	9
2.3.6 Entidad Profesor . . . . .	9
2.3.7 Entidad Secretario . . . . .	9
2.3.8 Entidad Director . . . . .	10
2.3.9 Entidad Consejero . . . . .	10
2.3.10 Entidad Tutor . . . . .	10
2.3.11 Entidad Sede . . . . .	10
2.3.12 Entidad Salon . . . . .	10
2.3.13 Entidad Grado . . . . .	10
2.3.14 Entidad Curso . . . . .	10
2.3.15 Entidad Matrícula . . . . .	10
<b>3 Modelo Relacional</b>	<b>11</b>
3.1 Modelo Relacional . . . . .	11
3.2 Especificaciones de transformación . . . . .	11
3.2.1 Entidades . . . . .	11
3.2.2 Entidades débiles . . . . .	12
3.2.3 Entidades superclases y subclases . . . . .	12
3.2.4 Relaciones binarias . . . . .	12
3.2.5 Relaciones ternarias . . . . .	12
3.3 Diccionario de datos . . . . .	12
<b>4 Implementación de la base de datos</b>	<b>16</b>
4.1 Creación de tablas en PostgreSQL . . . . .	16
4.2 Creación de triggers en PostgreSQL . . . . .	18
4.3 Creación de vistas en PostgreSQL . . . . .	22
4.4 Carga de datos . . . . .	23
4.5 Simulación de datos faltantes . . . . .	23
4.6 Generacion de gráficos . . . . .	23

<b>5 Optimización y experimentación</b>	<b>23</b>
5.1 Consultas SQL para el experimento . . . . .	24
5.1.1 Descripción del tipo de consultas seleccionadas . . . . .	24
5.1.2 Implementación de consultas en SQL . . . . .	24
5.2 Metodología del experimento . . . . .	26
5.3 Optimización de consultas . . . . .	26
5.3.1 Planes de índices para la primera consulta . . . . .	26
5.3.2 Planes de índices para la segunda consulta . . . . .	49
5.3.3 Planes de índices para la tercera consulta . . . . .	62
5.4 Plataforma de pruebas . . . . .	87
5.5 Medición de tiempos . . . . .	87
5.5.1 Sin índices . . . . .	87
5.5.2 Con índices por defecto . . . . .	88
5.5.3 Con índices por defecto más índices personalizados . . . . .	89
5.6 Resultados, análisis y discusión . . . . .	90
5.6.1 Consulta 1 . . . . .	92
5.6.2 Consulta 2 . . . . .	95
5.6.3 Consulta 3 . . . . .	98
<b>6 Conclusiones</b>	<b>99</b>
<b>7 Anexos</b>	<b>100</b>
7.1 Modelo Físico . . . . .	100
7.2 Repositorios de GitHub . . . . .	100
7.2.1 Repositorio del informe en LaTeX . . . . .	100
7.2.2 Repositorio del código en SQL . . . . .	100
7.2.3 Repositorio para el front-end . . . . .	101
7.3 Videos de experimentación . . . . .	101
7.3.1 Consulta 1 . . . . .	101
7.3.2 Consulta 2 . . . . .	101
7.3.3 Consulta 3 . . . . .	101
7.4 Pregunta extra . . . . .	101

# 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 pago 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 ajena 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 modular 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 secretaría 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 secretaría 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 proyecto

- 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

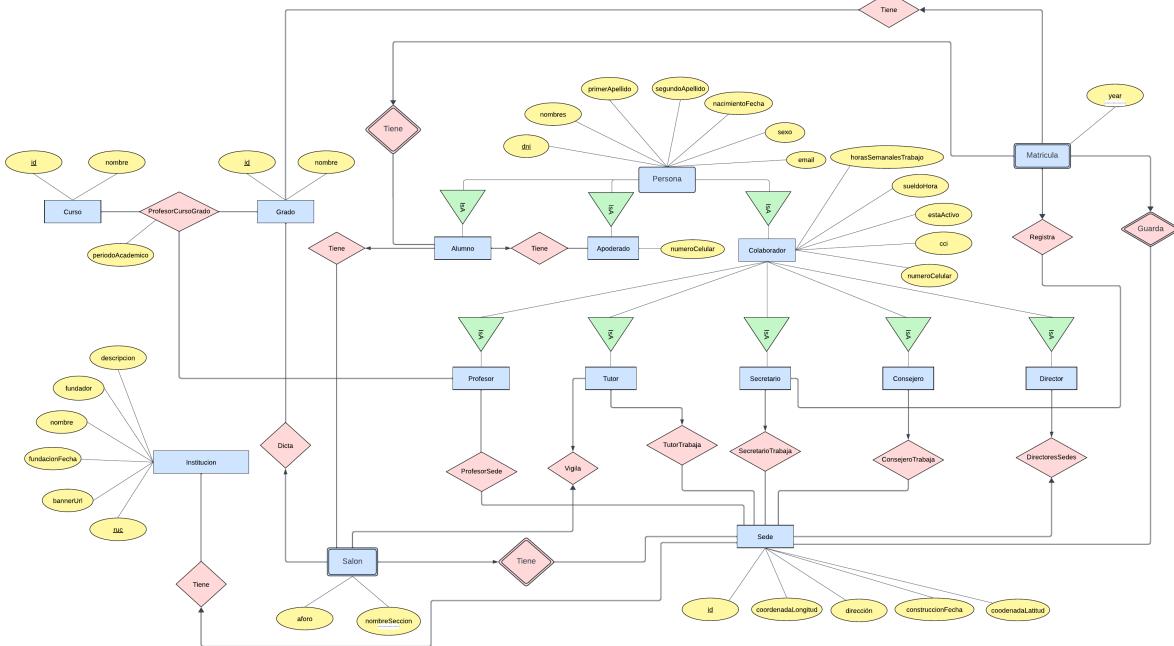


Figure 1: Modelo Entidad-Relación

## **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 Salón**

- 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 (dni),
32     sueldo_hora  FLOAT           NOT NULL,
33     cci          CHAR(20)         NOT NULL,
34     numero_celular VARCHAR(15)    NOT NULL,
35     horas_semanales_trabajo INT    NOT NULL,
36     esta_activo   BOOLEAN         NOT NULL,
37     CHECK (sueldo_hora > 0.0),
38     CHECK (cci NOT LIKE '%[^0-9]%' ),
39     CHECK (numero_celular LIKE '+%[0-9 ]%', OR numero_celular NOT LIKE '%[^0-9 ]%' ),
40     CHECK (horas_semanales_trabajo BETWEEN 1 AND 60)
41 );
42
43 CREATE TABLE sede
44 (
45     id           SERIAL PRIMARY KEY,
46     coordenada_longitud DOUBLE PRECISION NOT NULL,
47     coordenada_latitud  DOUBLE PRECISION NOT NULL,
48     direccion     VARCHAR(255)      NOT NULL,
49     construccion_fecha DATE          NOT NULL,
50     institucion_ruc CHAR(11) REFERENCES institucion (ruc) NOT 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 (dni),
104     salon_nombre_seccion VARCHAR(50) NOT NULL,
105     sede_id  INT          NOT NULL,
106     FOREIGN KEY (salon_nombre_seccion, sede_id) REFERENCES salon (nombre_seccion,
107     sede_id),
108     UNIQUE (salon_nombre_seccion, sede_id)
109 );
110
111 CREATE TABLE apoderado
112 (
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 LIKE '%[^0-9 ]%')
115 );
116
117 CREATE TABLE alumno
118 (
119     dni      CHAR(8) PRIMARY KEY REFERENCES persona (dni),
120     salon_nombre_seccion VARCHAR(50)          NOT NULL,
121     salon_sede_id    INT          NOT NULL,
122     apoderado_dni   CHAR(8) REFERENCES apoderado (dni) NOT NULL,
123     FOREIGN KEY (salon_nombre_seccion, salon_sede_id) REFERENCES salon (nombre_seccion,
124     sede_id)
125 );
126 CREATE TABLE profesor_sede
127 (

```

```

128     profesor_dni CHAR(8) REFERENCES profesor (dni),
129     sede_id        INT REFERENCES sede (id) NOT NULL,
130     PRIMARY KEY (profesor_dni, sede_id)
131 );
132
133 CREATE TABLE profesor_curso_grado
134 (
135     curso_id        INT REFERENCES curso (id),
136     grado_id        INT REFERENCES grado (id),
137     profesor_dni    CHAR(8) REFERENCES profesor (dni),
138     periodo_academico INT NOT NULL,
139     PRIMARY KEY (curso_id, grado_id, profesor_dni),
140     CHECK (periodo_academico <= EXTRACT(YEAR FROM CURRENT_DATE))
141 );
142
143 CREATE TABLE matricula
144 (
145     year            INT                                NOT NULL,
146     alumno_dni      CHAR(8) REFERENCES alumno (dni),
147     sede_id         INT REFERENCES sede (id),
148     grado_id        INT REFERENCES grado (id)          NOT NULL,
149     secretario_dni  CHAR(8) REFERENCES secretario (dni) NOT NULL,
150     PRIMARY KEY (year, alumno_dni, sede_id),
151     CHECK (year <= EXTRACT(YEAR FROM CURRENT_DATE))
152 );

```

## 4.2 Creación de triggers en PostgreSQL

```

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

```

```

42 CREATE OR REPLACE FUNCTION function_check_colaborador_overlapping() RETURNS TRIGGER AS
43 $$$
44 BEGIN
45     IF (EXISTS (SELECT 1 FROM profesor WHERE dni = new.dni)) OR
46         (EXISTS (SELECT 1 FROM consejero WHERE dni = new.dni)) OR
47         (EXISTS (SELECT 1 FROM secretario WHERE dni = new.dni)) OR
48         (EXISTS (SELECT 1 FROM director WHERE dni = new.dni)) OR
49         (EXISTS (SELECT 1 FROM tutor WHERE dni = new.dni)) THEN
50             RAISE EXCEPTION 'El colaborador con DNI % ya existe en otra tabla hija.', new.
51             dni;
52         END IF;
53
54     RETURN new;
55 END;
56 $$ LANGUAGE plpgsql;
57
58 CREATE TRIGGER trigger_check_profesor_overlapping
59     BEFORE INSERT
60     ON profesor
61     FOR EACH ROW
62 EXECUTE FUNCTION function_check_colaborador_overlapping();
63
64 CREATE TRIGGER trigger_check_consejero_overlapping
65     BEFORE INSERT
66     ON consejero
67     FOR EACH ROW
68 EXECUTE FUNCTION function_check_colaborador_overlapping();
69
70 CREATE TRIGGER trigger_check_secretario_overlapping
71     BEFORE INSERT
72     ON secretario
73     FOR EACH ROW
74 EXECUTE FUNCTION function_check_colaborador_overlapping();
75
76 CREATE TRIGGER trigger_check_director_overlapping
77     BEFORE INSERT
78     ON director
79     FOR EACH ROW
80 EXECUTE FUNCTION function_check_colaborador_overlapping();
81
82 CREATE TRIGGER trigger_check_tutor_overlapping
83     BEFORE INSERT
84     ON tutor
85     FOR EACH ROW
86 EXECUTE FUNCTION function_check_colaborador_overlapping();
87 -----
88
89 CREATE OR REPLACE FUNCTION function_check_colaborador_apoderado_overlapping_alumno()
90     RETURNS TRIGGER AS
91 $$$
92 BEGIN
93     IF EXISTS (SELECT 1 FROM alumno WHERE dni = new.dni) THEN
94         RAISE EXCEPTION 'La persona con DNI % ya existe en la tabla Alumno.', new.dni;
95     END IF;
96
97     RETURN new;
98 END;
99 $$ LANGUAGE plpgsql;
100
101 CREATE TRIGGER trigger_check_colaborador_overlapping
102     BEFORE INSERT
103     ON colaborador
104     FOR EACH ROW
105 EXECUTE FUNCTION function_check_colaborador_apoderado_overlapping_alumno();
106
107 CREATE TRIGGER trigger_check_apoderado_overlapping
108     BEFORE INSERT
109     ON apoderado
110     FOR EACH ROW
111 EXECUTE FUNCTION function_check_colaborador_apoderado_overlapping_alumno();
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 $$$
142 BEGIN
143     IF (SELECT COUNT(*) FROM director WHERE sede_id = old.sede_id AND dni != old.dni) =
144         0 THEN
145         RAISE EXCEPTION 'No se puede reasignar el director sin reemplazo en la sede %', old.sedeid;
146     END IF;
147
148     DELETE FROM director WHERE dni = old.dni;
149
150     UPDATE colaborador SET esta_activo = FALSE WHERE dni = old.dni;
151
152     RETURN new;
153 END;
154 $$ LANGUAGE plpgsql;
155
156 CREATE TRIGGER trigger_gestionar_director_reasignacion
157     AFTER UPDATE
158     ON director
159     FOR EACH ROW
160     WHEN (old.dni IS DISTINCT FROM new.dni)
161 EXECUTE FUNCTION gestionar_director_reasignacion();
162
163 -----
164
165 CREATE OR REPLACE FUNCTION function_check_matricula_year() RETURNS TRIGGER AS
166 $$$
167 BEGIN
168     IF new.year <= (SELECT EXTRACT(YEAR FROM construccion_fecha) FROM sede WHERE id =
169     new.sede_id) THEN
170         RAISE EXCEPTION 'El año de matrícula debe ser mayor que el año de construcción
171         de la sede.';
172     END IF;
173     RETURN new;
174 END;
175 $$ LANGUAGE plpgsql;
176
177 CREATE TRIGGER trigger_check_matricula_year
178     BEFORE INSERT OR UPDATE
179     ON matricula
180     FOR EACH ROW
181 EXECUTE FUNCTION function_check_matricula_year();
182
183 -----
184

```

```

182 CREATE OR REPLACE FUNCTION function_check_salon_aforo_on_alumno()
183     RETURNS TRIGGER AS
184 $$
185 DECLARE
186     salon_aforo      INT;
187     alumnos_cantidad INT;
188 BEGIN
189     SELECT aforo
190     INTO salon_aforo
191     FROM salon
192     WHERE nombre_seccion = new.nombre_seccion
193         AND sede_id = new.sede_id;
194
195     SELECT COUNT(dni) + 1
196     INTO alumnos_cantidad
197     FROM alumno
198     WHERE salon_nombre_seccion = new.salon_nombre_seccion
199         AND salon_sede_id = new.salon_sede_id;
200
201     IF alumnos_cantidad > salon_aforo THEN
202         RAISE EXCEPTION 'El aforo del salon ha sido excedido. Aforo maximo: %, Numero de
203         alumnos: %', salon_aforo, alumnos_cantidad;
204     END IF;
205
206     RETURN new;
207 END;
208 $$ LANGUAGE plpgsql;
209
210 CREATE TRIGGER trigger_check_salon_aforo_on_alumno
211     BEFORE INSERT
212     ON alumno
213     FOR EACH ROW
214     EXECUTE FUNCTION function_check_salon_aforo_on_alumno();
215
216 ----
217
218 CREATE OR REPLACE FUNCTION function_check_salon_aforo_on_matricula()
219     RETURNS TRIGGER AS
220 $$
221 DECLARE
222     salon_aforo      INT;
223     alumnos_cantidad INT;
224     salon_nombre_seccion VARCHAR(50);
225     salon_sede_id    INT;
226 BEGIN
227     SELECT salon_nombre_seccion, salon_sede_id
228     INTO salon_nombre_seccion, salon_sede_id
229     FROM alumno
230     WHERE dni = new.alumno_dni;
231
232     SELECT aforo
233     INTO salon_aforo
234     FROM salon
235     WHERE salon_nombre_seccion = new.nombre_seccion
236         AND salon_sede_id = new.sede_id;
237
238     SELECT COUNT(dni) + 1
239     INTO alumnos_cantidad
240     FROM alumno
241     WHERE salon_nombre_seccion = new.nombre_seccion
242         AND salon_sede_id = new.sede_id;
243
244     IF alumnos_cantidad > salon_aforo THEN
245         RAISE EXCEPTION 'El aforo del salon ha sido excedido. Aforo maximo: %, Numero de
246         alumnos: %', salon_aforo, alumnos_cantidad + 1;
247     END IF;
248
249     RETURN new;
250 END;
251 $$ LANGUAGE plpgsql;
252
253 CREATE TRIGGER trigger_check_salon_aforo_on_matricula
254     BEFORE INSERT

```

```

253     ON matricula
254     FOR EACH ROW
255 EXECUTE FUNCTION function_check_salon_aforo_on_matricula();
256
257 -----
258
259 CREATE OR REPLACE FUNCTION gestionar_tutor_reasignacion()
260 RETURNS TRIGGER AS
261 $$$
262 BEGIN
263     IF (SELECT COUNT(*)
264         FROM tutor
265         WHERE salon_nombre_seccion = old.salon_nombre_seccion
266             AND sede_id = old.sede_id
267             AND dni != old.dni) = 0 THEN
268         RAISE EXCEPTION 'No se puede reasignar el tutor sin reemplazo en el salon % de
269             la sede %', old.salon_nombre_seccion, old.sede_id;
270     END IF;
271
272     DELETE FROM tutor WHERE dni = old.dni AND salon_nombre_seccion = old.
273     salon_nombre_seccion AND sede_id = old.sede_id;
274
275     UPDATE colaborador SET esta_activo = FALSE WHERE dni = old.dni;
276
277     RETURN new;
278 END;
279 $$ LANGUAGE plpgsql;
280
281 CREATE TRIGGER trigger_gestionar_tutor_reasignacion
282     AFTER UPDATE
283     ON tutor
284     FOR EACH ROW
285     WHEN (old.dni IS DISTINCT FROM new.dni OR old.salon_nombre_seccion IS DISTINCT FROM
286             new.salon_nombre_seccion OR
287             old.sede_id IS DISTINCT FROM new.sede_id)
288 EXECUTE FUNCTION gestionar_tutor_reasignacion();

```

### 4.3 Creación de vistas en PostgreSQL

```

1 CREATE VIEW vista_colaborador_sueldo_mensual AS
2 SELECT
3     p.dni,
4     p.email,
5     c.cci,
6     c.est_activo,
7     (c.horas_semanales_trabajo * c.sueldo_hora * 4) AS sueldo_mensual
8 FROM
9     colaborador c
10    JOIN
11     persona p ON c.dni = p.dni;
12
13 -----
14
15 CREATE VIEW vista_profesor_curso_grado AS
16 SELECT
17     c.nombre AS curso_nombre,
18     g.nombre AS grado_nombre,
19     p.nombres AS profesor_nombres,
20     p.primer_apellido AS profesor_primer_apellido,
21     p.segundo_apellido AS profesor_segundo_apellido,
22     pcg.periodo_academico
23 FROM
24     curso c
25    JOIN
26     profesor_curso_grado pcg ON c.id = pcg.curso_id
27    JOIN
28     grado g ON pcg.grado_id = g.id
29    JOIN
30     profesor pr ON pcg.profesor_dni = pr.dni
31    JOIN
32     persona p ON pr.dni = p.dni;
33
34 -----

```

```

35
36 CREATE VIEW vista_total_alumnos_grado_sede AS
37 SELECT
38     s.direccion AS sede_direccion,
39     g.nombre AS grado_nombre,
40     COUNT(a.dni) AS total_alumnos
41 FROM
42     alumno a
43     JOIN
44     salon sa ON a.salon_nombre_seccion = sa.nombre_seccion AND a.salon_sede_id = sa.
45     sede_id
46     JOIN
47     grado g ON sa.grado_id = g.id
48     JOIN
49     sede s ON sa.sede_id = s.id
50 GROUP BY
51     g.id, g.nombre, s.id;
52 -----
53
54 CREATE VIEW vista_hijos_por_apoderado_sede AS
55 SELECT
56     ap.dni,
57     ap.numero_celular AS apoderado_numero_celular,
58     s.id AS sede_id,
59     COUNT(a.dni) AS total_hijos_en_sede
60 FROM
61     alumno a
62     JOIN
63     matricula m ON a.dni = m.alumno_dni
64     JOIN
65     apoderado ap ON a.apoderado_dni = ap.dni
66     JOIN
67     sede s ON m.sede_id = s.id
68 GROUP BY
69     ap.dni, ap.numero_celular, s.id;

```

#### 4.4 Carga de datos

Utilizamos Docker y Docker Compose para crear un contenedor en base a la imagen docker de PostgreSQL. En específico, Docker Compose nos ayudó a automatizar el proceso de creación de esquemas, tablas, disparadores y vistas, y la inserción de los datos faltantes en forma de CSV para cada esquema. Todo esto con el fin de facilitar y optimizar el trabajo colaborativo y la replicación del entorno de desarrollo.

#### 4.5 Simulación de datos faltantes

Para simular datos faltantes, se desarrolló un módulo 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.

#### 4.6 Generación de gráficos

La generación de diversos gráficos utilizados en nuestro proyecto requirieron el desarrollo de dos módulos en Python: uno para convertir los planes de ejecución en formato CSV a PNG, y otro para obtener las gráficas comparativas respecto a los índices para cada consulta. Para estos dos casos, utilizamos las librerías externas Pandas y Matplotlib.

### 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 excede 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 excede 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 los colaboradores que nacieron entre 1960 y 1980. 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.
  - **Justificación:** La institución educativa 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 de mayor edad, que trabajan en esa sede. 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 emitir 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 implementará la iniciativa de ofrecer un descuento especial en el pago mensual a los alumnos menores de edad cuyo apoderado sea un colaborador activo de la institución que labore a tiempo completo y su sueldo mensual 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 || ' ' || d_persona.
3     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 = d_colaborador.dni
13            JOIN persona AS d_persona ON d_colaborador.dni = d_persona.dni
14            JOIN sede ON director.sede_id = sede.id
15    WHERE
16        sede.construccion_fecha BETWEEN '1990-01-01' AND '2010-12-31'
```

```

16    AND (
17        SELECT COUNT(*)
18        FROM profesor_sede
19        WHERE profesor_sede.sede_id = sede.id
20    ) + (
21        SELECT COUNT(*)
22        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, ' ', persona.
2      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.horas_semanales_trabajo * 4 *
8                  0.05 *
9                  (SELECT COUNT(id)
10                 FROM sede
11                     JOIN profesor_sede ON sede.id = profesor_sede.sede_id
12                     WHERE profesor_sede.profesor_dni = profesor.dni
13                     AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM sede.
14                     construccion_fecha)) % 10 = 0
15                     AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM sede.
16                     construccion_fecha)) > 0)
17             ELSE
18                 colaborador.sueldo_hora * colaborador.horas_semanales_trabajo * 4 *
19                     0.05 *
20                     (CASE
21                         WHEN (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
22                             (SELECT MIN(sede.construccion_fecha)
23                             FROM sede JOIN colaborador AS c ON c.dni = colaborador.dni
24                             WHERE c.dni = colaborador.dni
25                             AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM sede.
26                             construccion_fecha)) % 10 = 0
27                             AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM sede.
28                             construccion_fecha)) > 0))) >= 10 THEN 1 ELSE 0 END)
29             END AS bonificacion
30         FROM colaborador
31             JOIN persona ON colaborador.dni = persona.dni
32             LEFT JOIN profesor ON colaborador.dni = profesor.dni
33         WHERE colaborador.está_activo = TRUE
34             AND persona.nacimiento_fecha BETWEEN '1960-01-01' AND '1980-12-31'
35             AND EXISTS (SELECT 1
36                 FROM sede JOIN colaborador AS c ON c.dni = colaborador.dni
37                 WHERE c.dni = colaborador.dni
38                     AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM sede.
39                     construccion_fecha)) % 10 = 0
40                     AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM sede.
41                     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 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 = TRUE
21                      AND colaborador.horas_semanales_trabajo > 48
22                          AND colaborador.sueldo_hora * colaborador.
23      horas_semanales_trabajo * 4 < 2000)))
24  AND persona.nacimiento_fecha > CURRENT_DATE - INTERVAL '18 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.

Sin embargo, antes de ejecutar las consultas sin índice y con el índice por defecto, es necesario ejecutar el comando **DROP INDEX IF EXISTS**. Esto asegura que cualquier índice personalizado creado previamente no interfiera con las consultas. De esta manera, garantizamos que las consultas se realicen correctamente y que los resultados reflejen el rendimiento de las consultas con y sin los índices predeterminados de la base de datos.

Por cada consulta realizada, se aplica un **VACUUM FULL** a todas las tablas involucradas para asegurar que estas se reestructuren completamente, recuperando espacio no utilizado y mejorando la eficiencia de la base de datos.

Finalmente, mediremos el tiempo de ejecución usando el comando **EXPLAIN ANALYZE** para probar el rendimiento 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

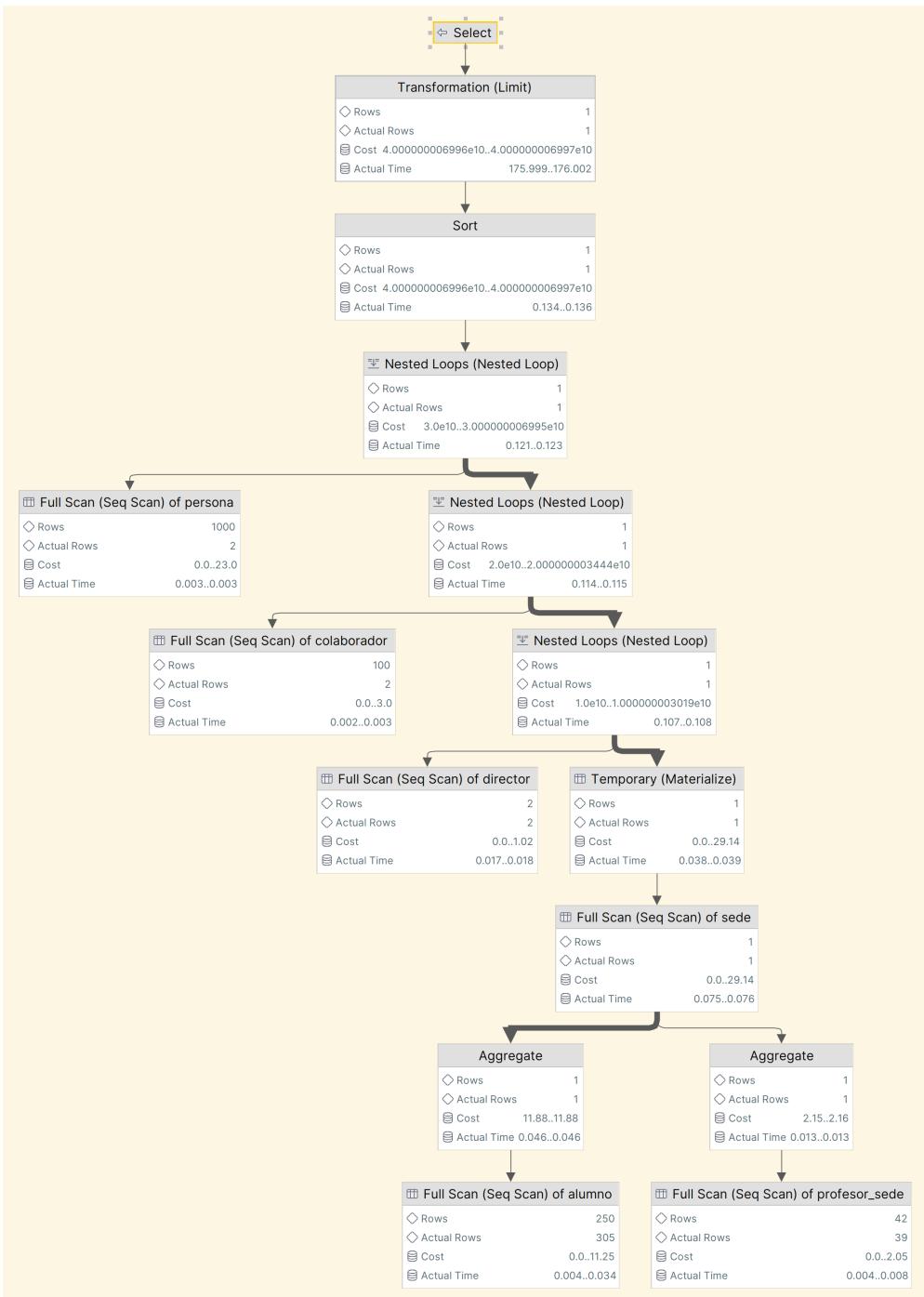
- Ejecución sin índices

```

1 SET enable_mergejoin TO 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 VACUUM FULL persona;
10 VACUUM FULL colaborador;
11 VACUUM FULL director;
12 VACUUM FULL sede;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL alumno;

```

– Para mil datos:



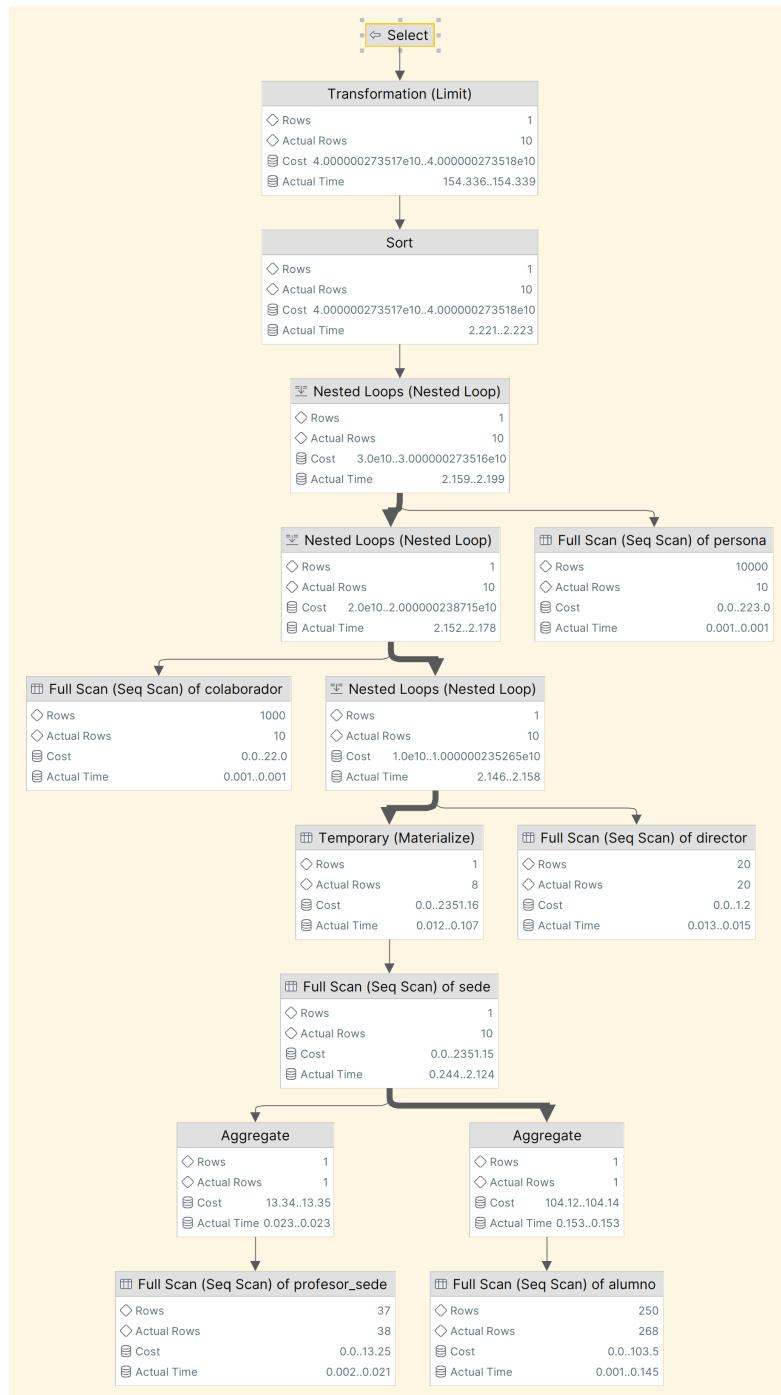
```

Limit  (cost=40000000069.96..40000000069.97 rows=1 width=600) (actual time=190.248..190.258 rows=1 loops=1)
-> Sort  (cost=40000000069.96..40000000069.97 rows=1 width=600) (actual time=0.120..0.122 rows=1 loops=1)
  Sort Key: sede.construcion_fecha
  Sort Method: quicksort Memory: 25kB
-> Nested Loop  (cost=30000000000.00..30000000069.95 rows=1 width=600) (actual time=0.105..0.107 rows=1 loops=1)
  Join Filter: (director.dni = d_persona.dni)
  Rows Removed by Join Filter: 1
-> Nested Loop  (cost=20000000000.00..20000000034.44 rows=1 width=591) (actual time=0.098..0.100 rows=1 loops=1)
  Join Filter: (director.dni = d_colaborador.dni)
  Rows Removed by Join Filter: 1
-> Nested Loop  (cost=10000000000.00..10000000030.19 rows=1 width=572) (actual time=0.091..0.093 rows=1 loops=1)
  Join Filter: (director.sede_id = sede.id)
  Rows Removed by Join Filter: 1
-> Seq Scan on director  (cost=0.00..1.02 rows=2 width=40) (actual time=0.012..0.013 rows=2 loops=1)
-> Materialize  (cost=0.00..29.14 rows=1 width=540) (actual time=0.037..0.037 rows=1 loops=2)
  -> Seq Scan on sede  (cost=0.00..29.14 rows=1 width=540) (actual time=0.072..0.073 rows=1 loops=1)
    Filter: ((construcion_fecha >= '1920-01-01':date) AND (construcion_fecha <= '2010-12-31':date) AND (((SubPlan 1) + (SubPlan 2)) <= 400))
    Rows Removed by Filter: 1
  SubPlan 1
    -> Aggregate  (cost=2.15..2.16 rows=1 width=8) (actual time=0.013..0.013 rows=1 loops=1)
      -> Seq Scan on profesor_sede  (cost=0.00..2.05 rows=42 width=0) (actual time=0.004..0.008 rows=39 loops=1)
        Filter: (sede_id = sede.id)
        Rows Removed by Filter: 45
  SubPlan 2
    -> Aggregate  (cost=11.88..11.88 rows=1 width=8) (actual time=0.045..0.045 rows=1 loops=1)
      -> Seq Scan on alumno  (cost=0.00..11.25 rows=250 width=0) (actual time=0.004..0.034 rows=305 loops=1)
        Filter: (salon_sede_id = sede.id)
        Rows Removed by Filter: 195
-> Seq Scan on colaborador d_colaborador  (cost=0.00..3.00 rows=100 width=19) (actual time=0.002..0.002 rows=2 loops=1)
-> Seq Scan on persona d_persona  (cost=0.00..23.00 rows=1000 width=59) (actual time=0.002..0.002 rows=2 loops=1)

Planning Time: 0.735 ms
JIT:
  Functions: 31
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.670 ms, Inlining 14.921 ms, Optimization 107.488 ms, Emission 67.725 ms, Total 191.804 ms
Execution Time: 192.028 ms

```

- Para diez mil datos:



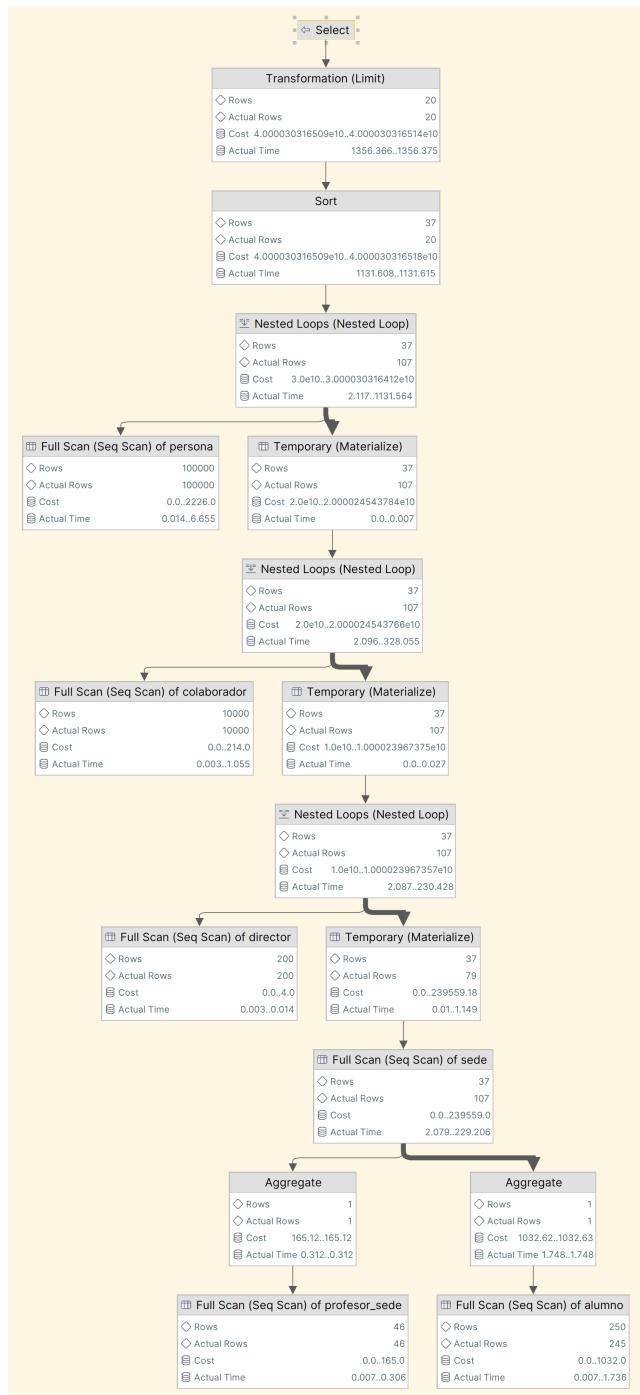
```

Limit  (cost=40000002735.17..40000002735.18 rows=1 width=600) (actual time=249.370..249.373 rows=1 loops=1)
-> Sort  (cost=40000002735.17..40000002735.18 rows=1 width=600) (actual time=2.841..2.844 rows=10 loops=1)
  Sort Key: sede.construccion_fecha
  Sort Method: quicksort Memory: 27kB
-> Nested Loop  (cost=300000000000.00..30000002735.16 rows=1 width=600) (actual time=2.781..2.828 rows=10 loops=1)
  Join Filter: (director.dni = d_persona.dni)
  Rows Removed By Join Filter: 94
-> Nested Loop  (cost=200000000000.00..20000002387.15 rows=1 width=591) (actual time=2.773..2.803 rows=10 loops=1)
  Join Filter: (director.dni = d_colaborador.dni)
  Rows Removed By Join Filter: 96
-> Nested Loop  (cost=100000000000.00..10000002352.65 rows=1 width=572) (actual time=2.764..2.779 rows=10 loops=1)
  Join Filter: (director.sede_id = sede.id)
  Rows Removed By Join Filter: 145
-> Seq Scan on director  (cost=0.00..1.20 rows=20 width=40) (actual time=0.013..0.015 rows=20 loops=1)
-> Materialize  (cost=0.00..2351.16 rows=1 width=540) (actual time=0.012..0.137 rows=8 loops=20)
-> Seq Scan on sede  (cost=0.00..2351.15 rows=1 width=540) (actual time=0.241..0.241..2.737 rows=10 loops=1)
  Filter: ((construccion_fecha >= '1920-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date) AND (((SubPlan 1) + (SubPlan 2)) <= 400))
  Rows Removed By Filter: 10
SubPlan 1
-> Aggregate  (cost=13.34..13.35 rows=1 width=8) (actual time=0.028..0.028 rows=1 loops=12)
-> Seq Scan on profesor_sede  (cost=0.00..13.25 rows=37 width=0) (actual time=0.002..0.026 rows=38 loops=12)
  Filter: (sede_id = sede.id)
  Rows Removed By Filter: 702
-> SubPlan 2
-> Aggregate  (cost=104.12..104.14 rows=1 width=8) (actual time=0.197..0.197 rows=1 loops=12)
-> Seq Scan on alumno  (cost=0.00..103.50 rows=250 width=0) (actual time=0.002..0.188 rows=268 loops=12)
  Filter: (salon_sede_id = sede.id)
  Rows Removed By Filter: 4732
-> Seq Scan on colaborador d_colaborador  (cost=0.00..22.00 rows=1000 width=19) (actual time=0.001..0.001 rows=10 loops=10)
-> Seq Scan on persona d_persona  (cost=0.00..223.00 rows=10000 width=59) (actual time=0.001..0.001 rows=10 loops=10)

Planning Time: 0.525 ms
JIT:
  Functions: 31
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.338 ms, Inlining 20.686 ms, Optimization 144.891 ms, Emission 80.962 ms, Total 247.876 ms
Execution Time: 250.850 ms

```

- Para cien mil datos:

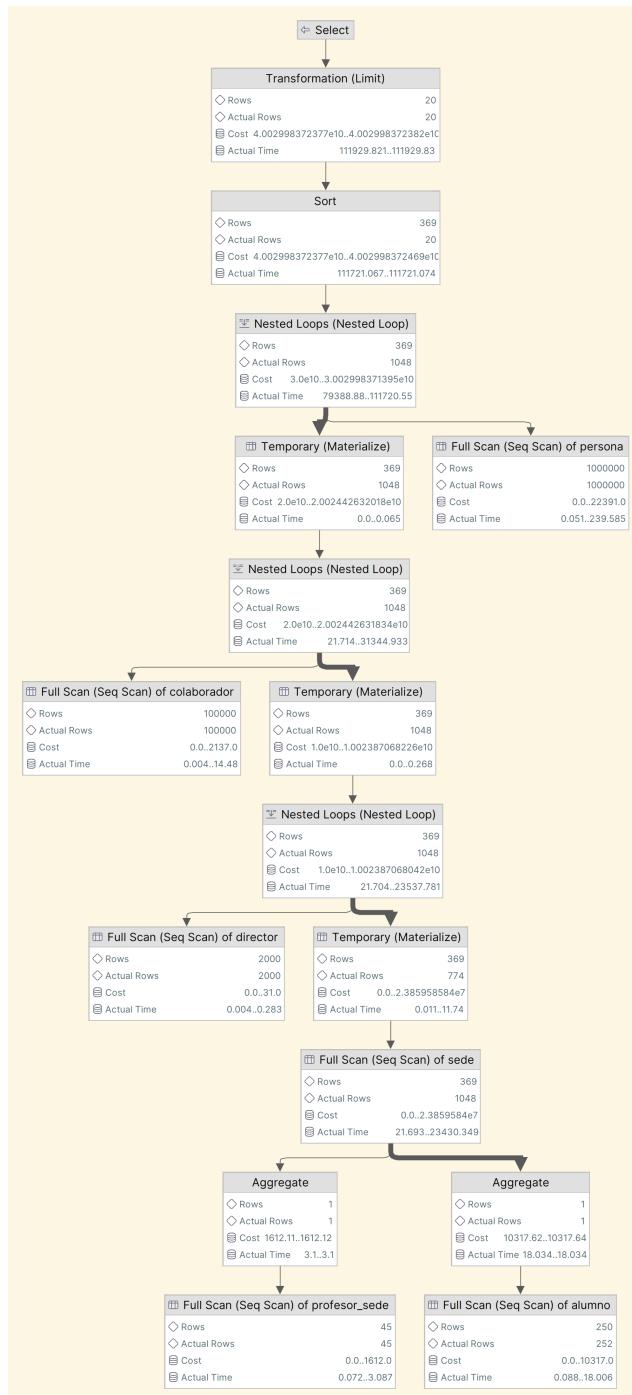


```

Limit  (cost=40000303165.09..40000303165.14 rows=20 width=129) (actual time=1528.882..1528.890 rows=20 loops=1)
-> Sort  (cost=40000303165.09..40000303165.18 rows=37 width=129) (actual time=1271.565..1271.571 rows=20 loops=1)
    Sort Key: sede.construccion_fecha
    Sort Method: top-N heapsort  Memory: 33kB
-> Nested Loop  (cost=3000000000.00..30000303164.12 rows=37 width=129) (actual time=2.167..1271.521 rows=107 loops=1)
    Join Filter: (director.dni = d_persona.dni)
    Rows Removed by Join Filter: 10699893
-> Seq Scan on persona d_persona  (cost=0.00..2226.00 rows=100000 width=60) (actual time=0.021..8.915 rows=100000 loops=1)
-> Materialize  (cost=2000000000.00..20000245437.84 rows=37 width=92) (actual time=0.000..0.007 rows=107 loops=100000)
    -> Nested Loop  (cost=2000000000.00..20000245437.66 rows=37 width=92) (actual time=2.139..329.917 rows=107 loops=1)
        Join Filter: (director.dni = d_colaborador.dni)
        Rows Removed by Join Filter: 1069893
-> Seq Scan on colaborador d_colaborador  (cost=0.00..214.00 rows=10000 width=19) (actual time=0.002..0.699 rows=10000 loops=1)
-> Materialize  (cost=1000000000.00..10000239673.75 rows=37 width=73) (actual time=0.000..0.028 rows=107 loops=10000)
    -> Nested Loop  (cost=1000000000.00..10000239673.57 rows=37 width=73) (actual time=2.131..243.657 rows=107 loops=1)
        Join Filter: (director.sede_id = sede.id)
        Rows Removed by Join Filter: 1069893
-> Seq Scan on director  (cost=0.00..4.00 rows=200 width=13) (actual time=0.004..0.021 rows=200 loops=1)
-> Materialize  (cost=0.00..239559.10 rows=37 width=68) (actual time=0.011..1.215 rows=79 loops=200)
    -> Seq Scan on sede  (cost=0.00..239559.00 rows=37 width=68) (actual time=2.121..241.828 rows=107 loops=1)
        Filter: ((construccion_fecha >= '1920-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date) AND (((SubPlan 1) + (SubPlan 2)) <=
        Rows Removed by Filter: 93
        SubPlan 1
            -> Aggregate  (cost=165.12..165.12 rows=1 width=8) (actual time=0.323..0.323 rows=1 loops=111)
                -> Seq Scan on profesor_sede  (cost=0.00..165.00 rows=46 width=0) (actual time=0.008..0.317 rows=46 loops=111)
                    Filter: (sede_id = sede.id)
                    Rows Removed by Filter: 9154
        Rows Removed by Filter: 9154
        SubPlan 2
            -> Aggregate  (cost=1032.62..1032.63 rows=1 width=8) (actual time=1.850..1.850 rows=1 loops=111)
                -> Seq Scan on alumno  (cost=0.00..1032.00 rows=250 width=0) (actual time=0.007..1.837 rows=245 loops=111)
                    Filter: (salon_sede_id = sede.id)
                    Rows Removed by Filter: 49755
Planning Time: 0.577 ms
JIT:
  Functions: 35
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.430 ms, Inlining 19.021 ms, Optimization 138.416 ms, Emission 99.900 ms, Total 258.767 ms
Execution Time: 1530.447 ms

```

– Para un millón de datos:



```

Limit (cost=40029983723.77..40029983723.82 rows=20 width=131) (actual time=110759.067..110759.076 rows=20 loops=1)
-> Sort (cost=40029983723.77..40029983724.69 rows=369 width=131) (actual time=110502.459..110502.467 rows=20 loops=1)
  Sort Key: sede.construccion_fecha
  Sort Method: top-N heapsort Memory: 34KB
-> Nested Loop (cost=3000000000.00..30029983713.95 rows=369 width=131) (actual time=77752.120..110502.064 rows=1048 loops=1)
  Join Filter: (director.dni = d_persona.dni)
  Rows Removed by Join Filter: 1047998952
-> Seq Scan on persona d_persona (cost=0.00..22391.00 rows=1000000 width=61) (actual time=0.081..216.767 rows=1000000 loops=1)
-> Materialize (cost=2000000000.00..20024426320.18 rows=369 width=93) (actual time=0.000..0.063 rows=1048 loops=1000000)
-> Nested Loop (cost=2000000000.00..20024426318.34 rows=369 width=93) (actual time=20.843..30135.045 rows=1048 loops=1)
  Join Filter: (director.dni = d_colaborador.dni)
  Rows Removed by Join Filter: 1047998952
-> Seq Scan on colaborador d_colaborador (cost=0.00..2137.00 rows=100000 width=19) (actual time=0.004..13.091 rows=100000 loops=1)
-> Materialize (cost=1000000000.00..10023870682.26 rows=369 width=74) (actual time=0.000..0.255 rows=1048 loops=100000)
-> Nested Loop (cost=1000000000.00..10023870680.42 rows=369 width=74) (actual time=20.831..22215.573 rows=1048 loops=1)
  Join Filter: (director.sede_id = sede.id)
  Rows Removed by Join Filter: 1546324
-> Seq Scan on director (cost=0.00..31.00 rows=2000 width=13) (actual time=0.004..0.259 rows=2000 loops=1)
-> Materialize (cost=0.00..23859585.84 rows=369 width=69) (actual time=0.010..11.079 rows=774 loops=2000)
-> Seq Scan on sede (cost=0.00..23859584.00 rows=369 width=69) (actual time=20.818..22110.626 rows=1048 loops=1)
  Filter: ((construccion_fecha >= '1920-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date) AND ((SubPlan 1) + (SubPlan 2)) <=
  Rows Removed by Filter: 952
  SubPlan 1
    -> Aggregate (cost=1612.11..1612.12 rows=1 width=8) (actual time=2.945..2.945 rows=1 loops=1108)
      -> Seq Scan on profesor_sede (cost=0.00..1612.00 rows=45 width=0) (actual time=0.069..2.934 rows=45 loops=1108)
        Filter: (sede_id = sede.id)
        Rows Removed by Filter: 89955
  SubPlan 2
    -> Aggregate (cost=10317.62..10317.64 rows=1 width=8) (actual time=16.997..16.997 rows=1 loops=1108)
      -> Seq Scan on alumno (cost=0.00..10317.00 rows=250 width=0) (actual time=0.081..16.972 rows=252 loops=1108)
        Filter: (salon_sede_id = sede.id)
        Rows Removed by Filter: 499748
Planning Time: 0.938 ms
JIT:
  Functions: 35
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 0.930 ms, Inlining 42.138 ms, Optimization 115.887 ms, Emission 98.653 ms, Total 257.609 ms
Timing: Generation 0.930 ms, Inlining 42.138 ms, Optimization 115.887 ms, Emission 98.653 ms, Total 257.609 ms
Execution Time: 110794.072 ms

```

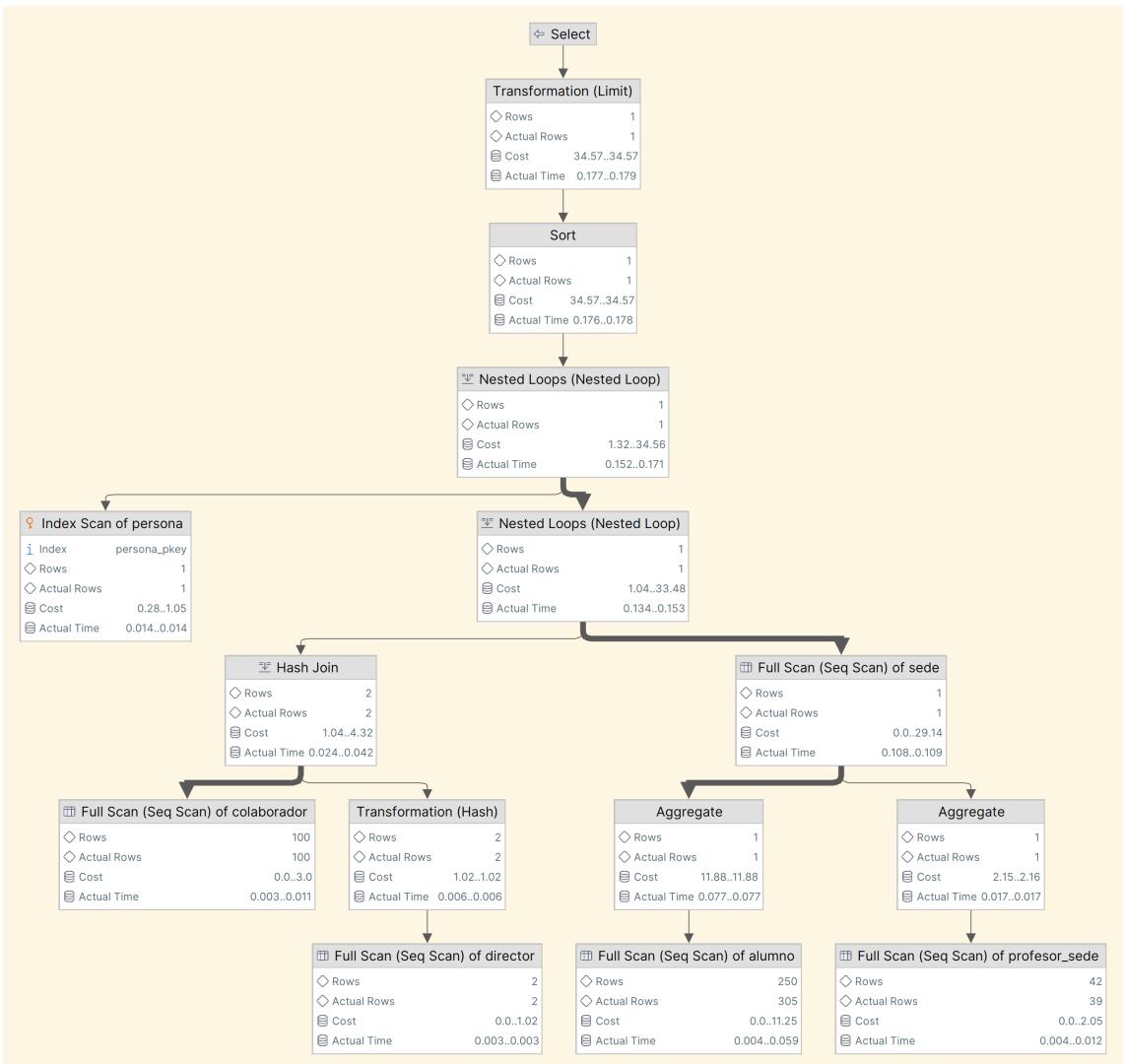
- Ejecución con índices por defecto

```

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_sede_construccion_fecha;
10
11 VACUUM FULL persona;
12 VACUUM FULL colaborador;
13 VACUUM FULL director;
14 VACUUM FULL sede;
15 VACUUM FULL profesor_sede;
16 VACUUM FULL alumno;

```

– Para mil datos:

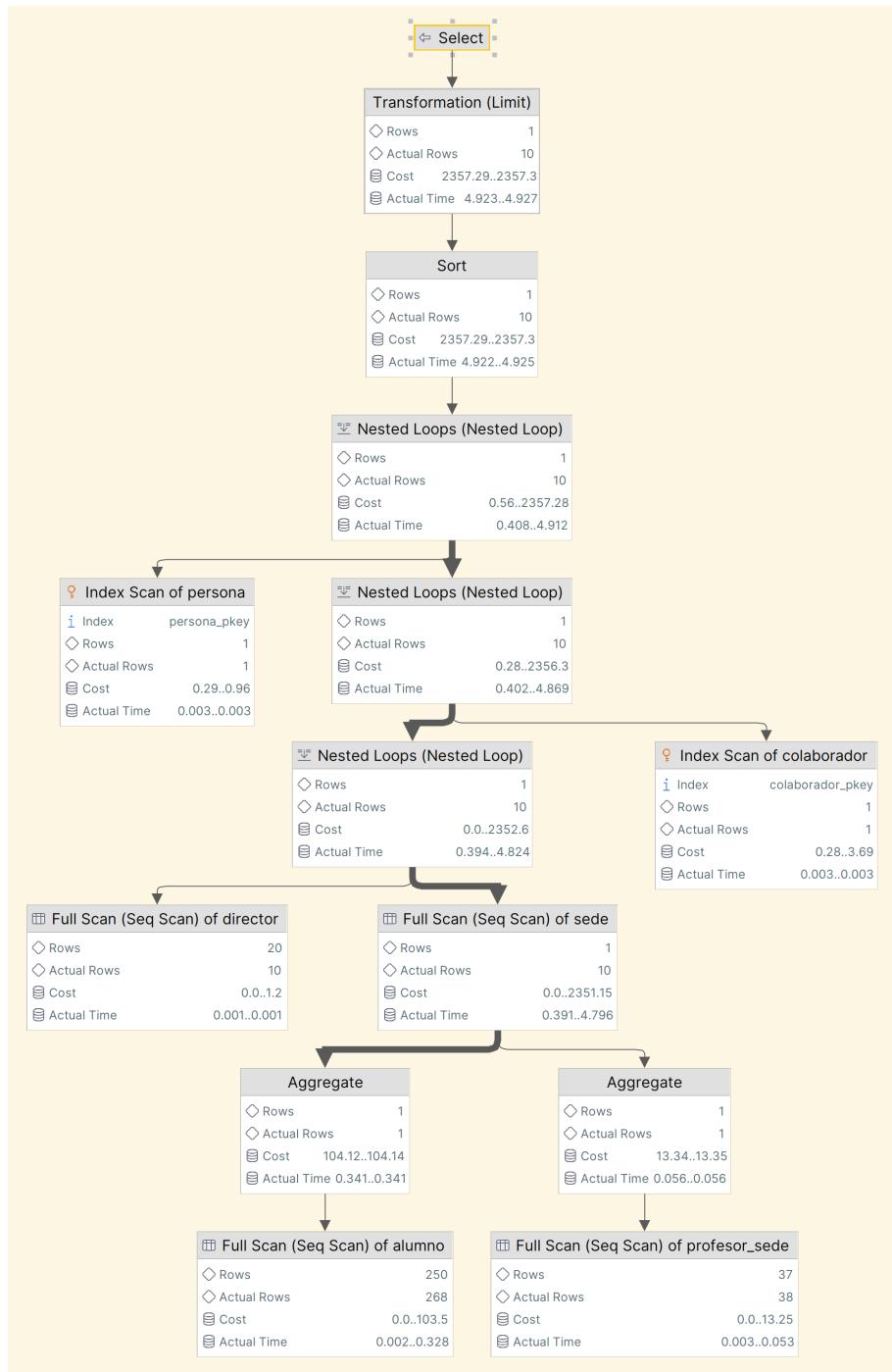


```

Limit (cost=34.57..34.57 rows=1 width=600) (actual time=0.176..0.173 rows=1 loops=1)
-> Sort (cost=34.57..34.57 width=600) (actual time=0.169..0.171 rows=1 loops=1)
  Sort Key: sede.construccion_fecha
  Sort Method: quicksort  Memory: 25kB
-> Nested Loop (cost=1.32..34.56 rows=1 width=600) (actual time=0.141..0.161 rows=1 loops=1)
  Join Filter: (director.dni = d.persona_dni)
-> Nested Loop (cost=1.04..33.48 rows=1 width=591) (actual time=0.126..0.145 rows=1 loops=1)
  Join Filter: (director.sede_id = sede.id)
  Row Removed by Join Filter: 1
-> Seq Scan on sede (cost=0.0..29.14 rows=1 width=540) (actual time=0.109..0.110 rows=1 loops=1)
  Filter: ((construccion_fecha >= '1990-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date) AND (((SubPlan 1) + (SubPlan 2)) <= 400))
  Rows Removed by Filter: 1
  SubPlan 1
    -> Aggregate (cost=2.15..2.16 rows=1 width=8) (actual time=0.020..0.020 rows=1 loops=1)
      -> Seq Scan on profesor_sede (cost=0.0..2.05 rows=42 width=0) (actual time=0.008..0.015 rows=39 loops=1)
        Filter: (sede_id = sede.id)
        Rows Removed by Filter: 45
    -> Aggregate (cost=11.88..11.88 rows=1 width=8) (actual time=0.071..0.071 rows=1 loops=1)
      -> Seq Scan on alumno (cost=0.0..11.25 rows=250 width=0) (actual time=0.004..0.055 rows=305 loops=1)
        Filter: (salon_sede_id = sede.id)
        Rows Removed by Filter: 195
  SubPlan 2
    -> Hash Join (cost=1.04..4.32 rows=2 width=59) (actual time=0.015..0.034 rows=2 loops=1)
      Hash Cond: (d.colaborador.dni = director.dni)
      -> Seq Scan on colaborador d_colaborador (cost=0.0..3.00 rows=100 width=19) (actual time=0.003..0.011 rows=100 loops=1)
      -> Hash (cost=1.02..1.02 rows=2 width=40) (actual time=0.005..0.005 rows=2 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 9kB
      -> Seq Scan on director (cost=0.0..1.02 rows=2 width=40) (actual time=0.002..0.002 rows=2 loops=1)
      Index Scan using persona_pkey on persona d_persona (cost=0.28..1.05 rows=1 width=59) (actual time=0.011..0.011 rows=1 loops=1)
      Index Cond: (dni = d.colaborador.dni)
Planning Time: 0.892 ms
Execution Time: 0.217 ms

```

– Para diez mil datos:

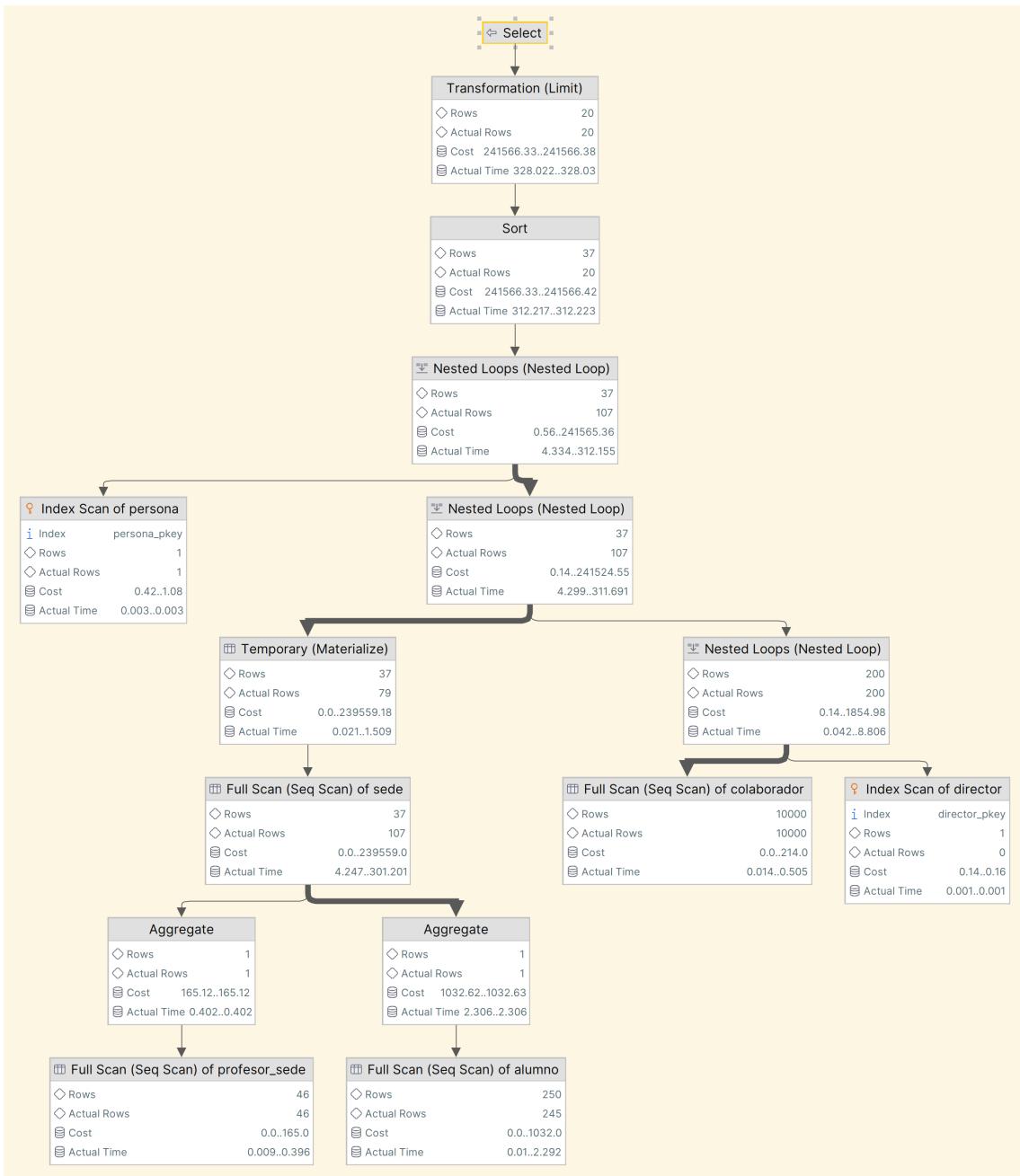


```

Limit (cost=2357.29..2357.30 rows=1 width=600) (actual time=5.001..5.006 rows=10 loops=1)
-> Sort (cost=2357.29..2357.30 rows=1 width=600) (actual time=5.000..5.004 rows=10 loops=1)
  Sort Key: sede.construccion_fecha
  Sort Method: quicksort  Memory: 27kB
-> Nested Loop (cost=0.56..2357.28 rows=1 width=600) (actual time=0.436..4.990 rows=10 loops=1)
  Join Filter: (director.dni = d_persona.dni)
-> Nested Loop (cost=0.28..2356.30 rows=1 width=591) (actual time=0.426..4.926 rows=10 loops=1)
  Join Filter: (director.sede_id = sede.id)
  Rows Removed by Join Filter: 94
-> Seq Scan on sede (cost=0.00..2351.15 rows=1 width=540) (actual time=0.414..4.834 rows=10 loops=1)
  Filter: ((construccion_fecha >= '1990-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date) AND (((SubPlan 1) + (SubPlan 2)) <= 400))
  Rows Removed by Filter: 10
-> SubPlan 1
  -> Aggregate (cost=13.34..13.35 rows=1 width=8) (actual time=0.052..0.052 rows=1 loops=12)
    -> Seq Scan on profesor_sede (cost=0.00..13.25 rows=37 width=0) (actual time=0.003..0.049 rows=38 loops=12)
      Filter: (sede_id = sede.id)
      Rows Removed by Filter: 702
-> SubPlan 2
  -> Aggregate (cost=104.12..104.14 rows=1 width=8) (actual time=0.349..0.349 rows=1 loops=12)
    -> Seq Scan on alumno (cost=0.00..103.50 rows=250 width=0) (actual time=0.002..0.337 rows=268 loops=12)
      Filter: (salon_sede_id = sede.id)
      Rows Removed by Filter: 4732
-> Seq Scan on director (cost=0.00..1.20 rows=20 width=40) (actual time=0.001..0.001 rows=10 loops=10)
-> Index Scan using colaborador_pkey on colaborador d_colaborador (cost=0.28..3.69 rows=1 width=19) (actual time=0.005..0.005 rows=1 loops=10)
  Index Cond: (dni = director.dni)
-> Index Scan using persona_pkey on persona d_persona (cost=0.29..0.96 rows=1 width=59) (actual time=0.005..0.005 rows=1 loops=10)
  Index Cond: (dni = d_colaborador.dni)
Planning Time: 0.754 ms
Execution Time: 5.048 ms

```

- Para cien mil datos:

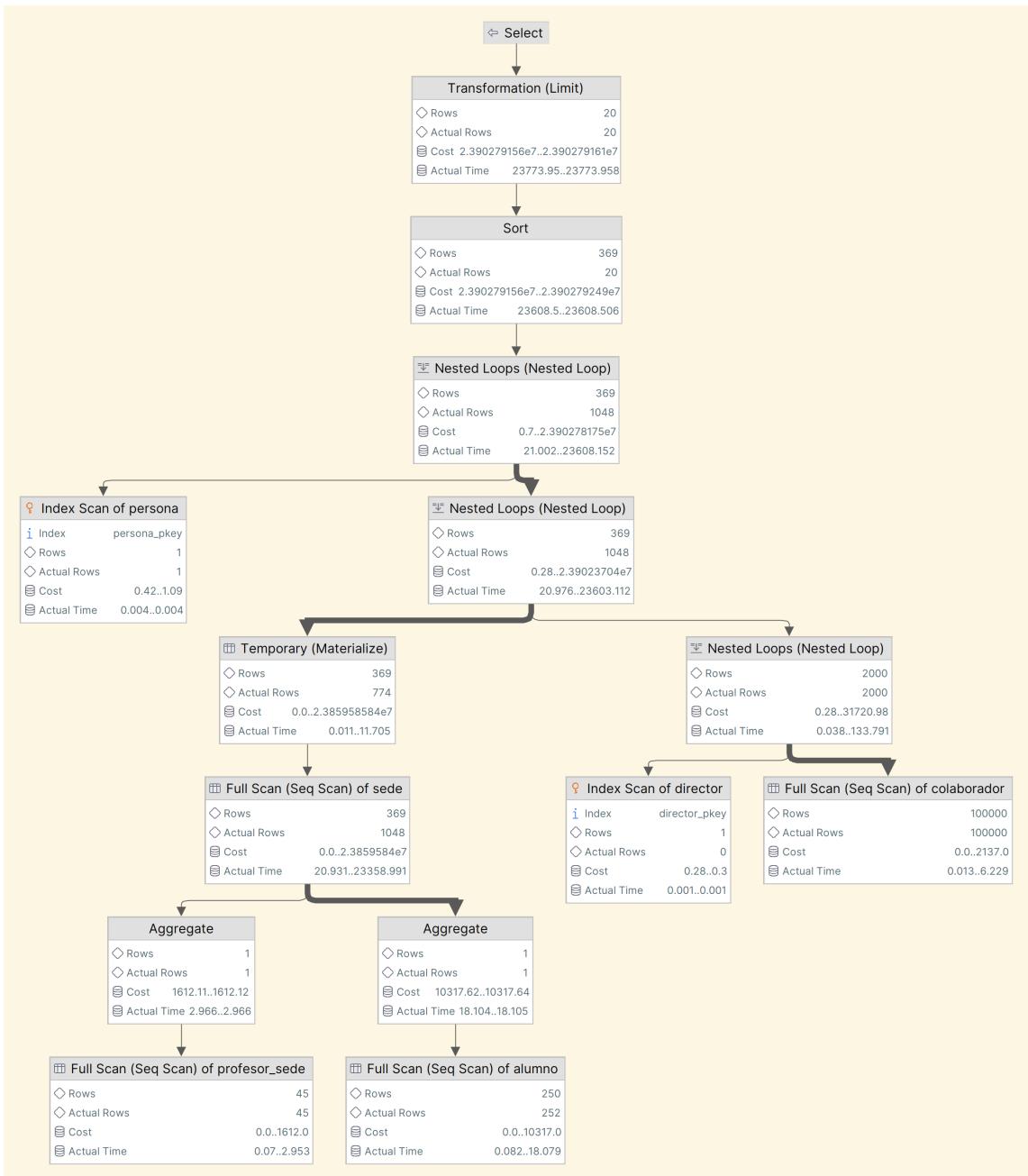


```

Limit (cost=241566.33..241566.38 rows=20 width=129) (actual time=378.532..378.541 rows=20 loops=1)
-> Sort (cost=241566.33..241566.42 rows=37 width=129) (actual time=357.627..357.634 rows=20 loops=1)
  Sort Key: sede.construccion_fecha
  Sort Method: top-N heapsort Memory: 33kB
-> Nested Loop (cost=0.56..241565.36 rows=37 width=129) (actual time=7.781..357.531 rows=107 loops=1)
  Join Filter: (director.dni = d_persona.dni)
-> Nested Loop (cost=0.14..241524.55 rows=37 width=92) (actual time=7.746..356.840 rows=107 loops=1)
  Join Filter: (director.sede_id = sede.id)
  Rows Removed by Join Filter: 15622
-> Nested Loop (cost=0.14..1854.98 rows=200 width=32) (actual time=4.251..13.051 rows=200 loops=1)
  -> Seq Scan on colaborador d_colaborador (cost=0.00..214.00 rows=10000 width=19) (actual time=0.013..0.524 rows=10000 loops=1)
  -> Index Scan using director_pkey on director (cost=0.14..0.16 rows=1 width=13) (actual time=0.001..0.001 rows=10000)
    Index Cond: (dni = d_colaborador.dni)
-> Materialize (cost=0.00..239559.18 rows=37 width=68) (actual time=0.017..1.713 rows=79 loops=200)
  -> Seq Scan on sede (cost=0.00..239559.00 rows=37 width=68) (actual time=3.485..341.936 rows=107 loops=1)
    Filter: ((construccion_fecha >= '1990-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date) AND (((SubPlan 1) + (SubPlan 2)) <= 400))
  Rows Removed by Filter: 93
  SubPlan 1
    -> Aggregate (cost=165.12..165.12 rows=1 width=8) (actual time=0.433..0.434 rows=1 loops=111)
      -> Seq Scan on profesor_sede (cost=0.00..165.00 rows=46 width=0) (actual time=0.009..0.425 rows=46 loops=111)
        Filter: (salon_sede_id = sede.id)
        Rows Removed by Filter: 9154
  SubPlan 2
    -> Aggregate (cost=1032.62..1032.63 rows=1 width=8) (actual time=2.639..2.640 rows=1 loops=111)
      -> Seq Scan on alumno (cost=0.00..1032.00 rows=250 width=0) (actual time=0.011..2.621 rows=245 loops=111)
        Filter: (salon_sede_id = sede.id)
        Rows Removed by Filter: 49755
-> Index Scan using persona_pkey on persona d_persona (cost=0.42..1.08 rows=1 width=60) (actual time=0.005..0.005 rows=1 loops=107)
  Index Cond: (dni = d_colaborador.dni)
Planning Time: 1.336 ms
JIT:
  Functions: 34
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 1.141 ms, Inlining 0.000 ms, Optimization 0.508 ms, Emission 24.624 ms, Total 26.273 ms
Execution Time: 379.886 ms

```

- Para un millón de datos:



```

Limit (cost=23902791.56..23902791.61 rows=20 width=131) (actual time=25310.885..25310.893 rows=20 loops=1)
-> Sort (cost=23902791.56..23902792.49 rows=369 width=131) (actual time=25124.945..25124.951 rows=20 loops=1)
  Sort Key: sede.construccion_fecha
  Sort Method: top-N heapsort Memory: 34KB
-> Nested Loop (cost=0.70..23902781.75 rows=369 width=131) (actual time=20.999..25124.576 rows=1048 loops=1)
  Join Filter: (director.dni = d_persona.dni)
-> Nested Loop (cost=0.28..23902370.40 rows=369 width=93) (actual time=20.960..25118.554 rows=1048 loops=1)
  Join Filter: (director.sede_id = sede.id)
  Rows Removed by Join Filter: 1546324
-> Nested Loop (cost=0.28..31720.98 rows=2000 width=32) (actual time=0.065..129.089 rows=2000 loops=1)
  -> Seq Scan on colaborador d_colaborador (cost=0.00..2137.00 rows=100000 width=19) (actual time=0.014..5.534 rows=100000 loops=1)
  -> Index Scan using director_pkey on director (cost=0.28..0.30 rows=1 width=13) (actual time=0.001..0.001 rows=100000)
    Index Cond: (dni = d_colaborador.dni)
-> Materialize (cost=0.00..23859585.84 rows=369 width=69) (actual time=0.010..12.464 rows=774 loops=2000)
  -> Seq Scan on sede (cost=0.00..23859584.00 rows=369 width=69) (actual time=20.887..24875.205 rows=1048 loops=1)
    Filter: ((construccion_fecha >= '1990-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date) AND (((SubPlan 1) + (SubPlan 2)) <= 400))
  Rows Removed by Filter: 952
  SubPlan 1
    -> Aggregate (cost=1612.11..1612.12 rows=1 width=8) (actual time=3.302..3.302 rows=1 loops=1108)
      -> Seq Scan on profesor_sede (cost=0.00..1612.00 rows=45 width=0) (actual time=0.077..3.288 rows=45 loops=1108)
        Filter: (sede_id = sede.id)
        Rows Removed by Filter: 89955
  SubPlan 2
    -> Aggregate (cost=10317.62..10317.64 rows=1 width=8) (actual time=19.135..19.135 rows=1 loops=1108)
      -> Seq Scan on alumno (cost=0.00..10317.00 rows=250 width=0) (actual time=0.085..19.105 rows=252 loops=1108)
        Filter: (salon_sede_id = sede.id)
        Rows Removed by Filter: 499748
  -> Index Scan using persona_pkey on persona d_persona (cost=0.42..1.09 rows=1 width=61) (actual time=0.005..0.005 rows=1 loops=1048)
    Index Cond: (dni = d_colaborador.dni)
Planning Time: 0.824 ms
JIT:
  Functions: 34
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.018 ms, Inlining 16.752 ms, Optimization 98.120 ms, Emission 71.121 ms, Total 187.011 ms
Execution Time: 25312.110 ms

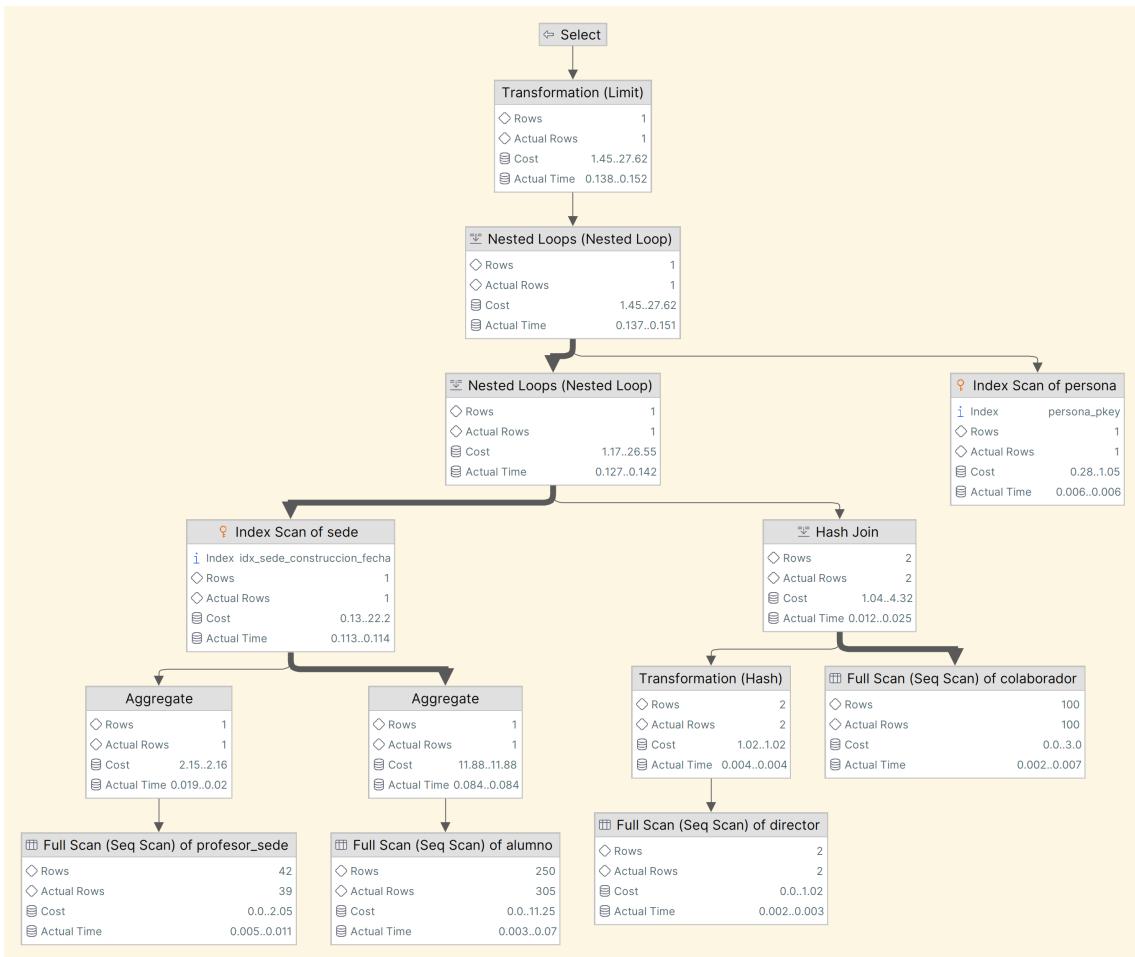
```

- Ejecución con índices por defecto más índices personalizados

```

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 VACUUM FULL persona;
10 VACUUM FULL colaborador;
11 VACUUM FULL director;
12 VACUUM FULL sede;
13 VACUUM FULL profesor_sede;
14 VACUUM FULL alumno;
15
16 CREATE INDEX IF NOT EXISTS idx_sede_construccion_fecha ON sede (construccion_fecha)
;
```

- Para mil datos:

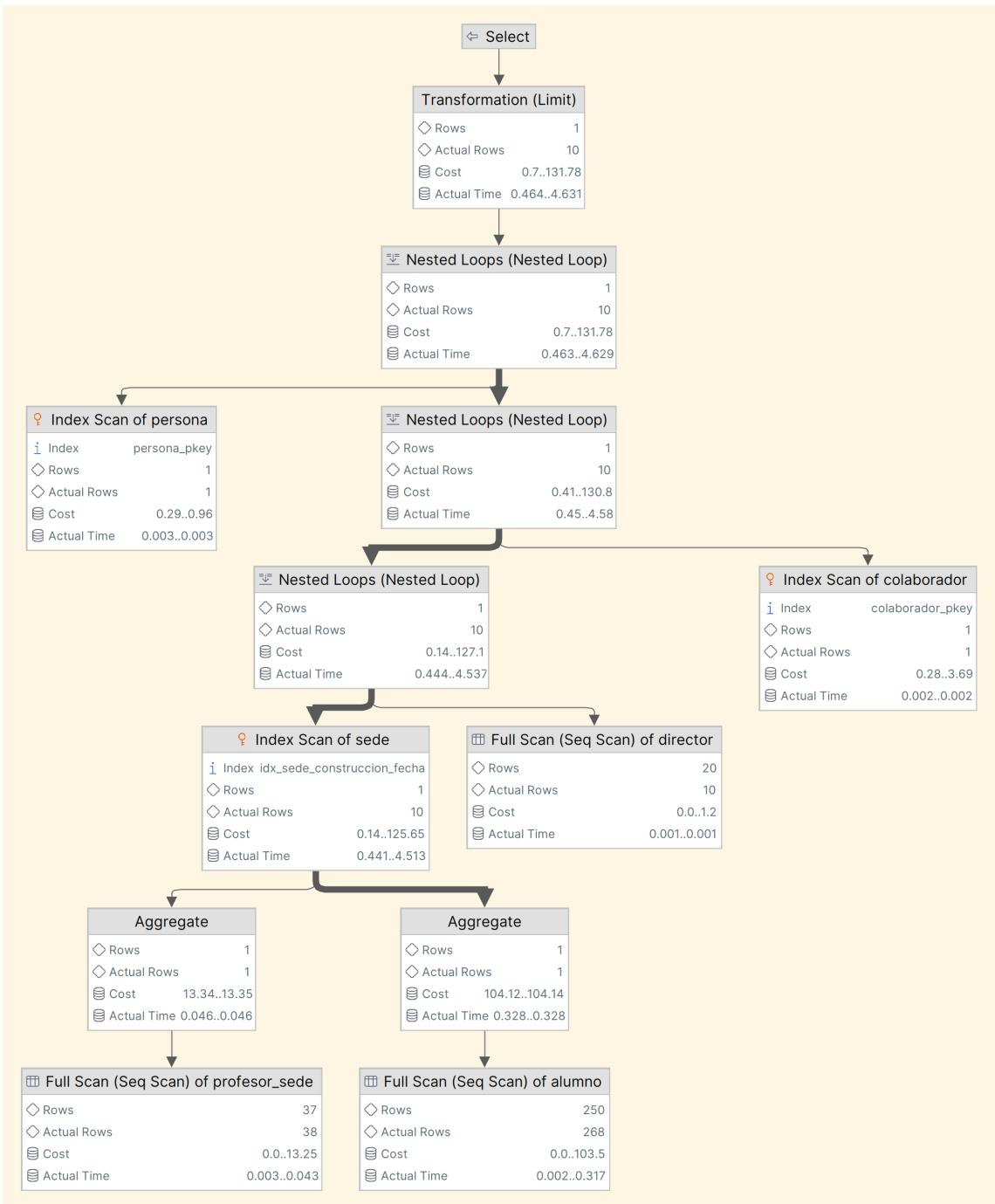


```

Limit (cost=1.45..27.62 rows=1 width=600) (actual time=0.102..0.116 rows=1 loops=1)
-> Nested Loop (cost=1.45..27.62 rows=1 width=600) (actual time=0.101..0.115 rows=1 loops=1)
  Join Filter: (director.dni = d_persona.dni)
  -> Nested Loop (cost=1.17..26.55 rows=1 width=591) (actual time=0.092..0.106 rows=1 loops=1)
    Join Filter: (director.sede_id = sede.id)
    Rows Removed by Join Filter: 1
    -> Index Scan using idx_sede_construccion_fecha on sede (cost=0.13..22.20 rows=1 width=540) (actual time=0.074..0.075 rows=1 loops=1)
      Index Cond: ((construccion_fecha >= '1990-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date))
      Filter: (((SubPlan 1) + (SubPlan 2)) <= 400)
      SubPlan 1
        -> Aggregate (cost=2.15..2.16 rows=1 width=8) (actual time=0.014..0.014 rows=1 loops=1)
          -> Seq Scan on profesor_sede (cost=0.00..2.05 rows=42 width=8) (actual time=0.005..0.010 rows=39 loops=1)
            Filter: (sede_id = sede.id)
            Rows Removed by Filter: 45
      SubPlan 2
        -> Aggregate (cost=11.88..11.88 rows=1 width=8) (actual time=0.050..0.051 rows=1 loops=1)
          -> Seq Scan on alumno (cost=0.00..11.25 rows=250 width=8) (actual time=0.003..0.039 rows=305 loops=1)
            Filter: (salon_sede_id = sede.id)
            Rows Removed by Filter: 195
    -> Hash Join (cost=1.04..4.32 rows=2 width=59) (actual time=0.017..0.029 rows=2 loops=1)
      Hash Cond: (d_colaborador.dni = director.dni)
      -> Seq Scan on colaborador_d_colaborador (cost=0.00..3.00 rows=100 width=19) (actual time=0.002..0.007 rows=100 loops=1)
      -> Hash (cost=1.02..1.02 rows=2 width=40) (actual time=0.003..0.003 rows=2 loops=1)
        Buckets: 1024 Batches: 1 Memory Usage: 9KB
        -> Seq Scan on director (cost=0.00..1.02 rows=2 width=40) (actual time=0.001..0.001 rows=2 loops=1)
      Index Cond: (dni = d_colaborador.dni)
Planning Time: 0.653 ms
Execution Time: 0.159 ms

```

– Para diez mil datos:



```

Limit (cost=0.70..131.78 rows=1 width=600) (actual time=0.568..4.762 rows=10 loops=1)
  -> Nested Loop (cost=0.70..131.78 rows=1 width=600) (actual time=0.567..4.759 rows=10 loops=1)
    Join Filter: (director.dni = d_persona.dni)
      -> Nested Loop (cost=0.41..130.80 rows=1 width=591) (actual time=0.551..4.695 rows=10 loops=1)
        Join Filter: (director.sede_id = sede.id)
          -> Nested Loop (cost=0.14..127.10 rows=1 width=572) (actual time=0.535..4.647 rows=10 loops=1)
            Join Filter: (director.sede_id = sede.id)
            Rows Removed by Join Filter: 94
            -> Index Scan using idx_sede_construcion_fecha on sede (cost=0.14..125.65 rows=1 width=540) (actual time=0.530..4.623 rows=10 loops=1)
              Index Cond: ((construcion.fecha >= '1990-01-01'::date) AND (construcion.fecha <= '2010-12-31'::date))
              Filter: (((SubPlan 1) + (SubPlan 2)) <= 400)
              Rows Removed by Filter: 2
              Rows Planed 1
              -> Aggregate (cost=13.34..13.35 rows=1 width=8) (actual time=0.050..0.050 rows=1 loops=12)
                -> Seq Scan on profesor_sede (cost=0.00..13.25 rows=37 width=0) (actual time=0.003..0.046 rows=38 loops=12)
                  Filter: (salon.sede_id = sede.id)
                  Rows Removed by Filter: 702
                  SubPlan 2
                  -> Aggregate (cost=104.12..104.14 rows=1 width=8) (actual time=0.332..0.332 rows=1 loops=12)
                    -> Seq Scan on alumno (cost=0.00..103.50 rows=250 width=0) (actual time=0.003..0.320 rows=268 loops=12)
                      Filter: (salon.sede_id = sede.id)
                      Rows Removed by Filter: 4732
                      -> Seq Scan on director (cost=0.00..1.20 rows=20 width=40) (actual time=0.001..0.001 rows=10 loops=10)
                      Index Scan using colaborador_pkey on colaborador d_colaborador (cost=0.20..3.69 rows=1 width=19) (actual time=0.004..0.004 rows=1 loops=10)
                      Index Cond: (dni = director.dni)
                      Index Cond: (dni = d_colaborador.dni)
Planning Time: 0.997 ms
Execution Time: 4.811 ms

```

- Para cien mil datos:

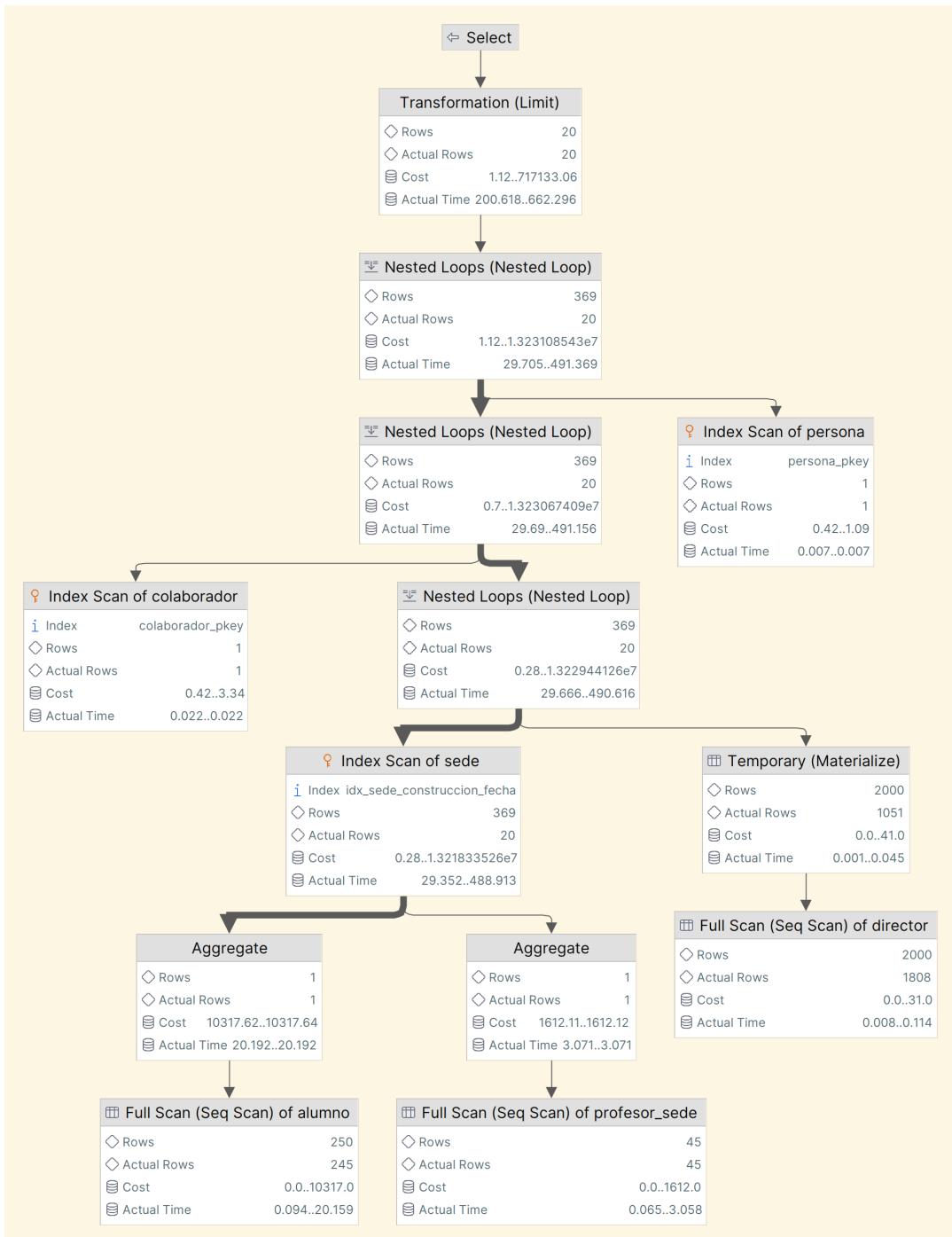


```

Limit (cost=0.85..72675.57 rows=20 width=129) (actual time=3.785..81.106 rows=20 loops=1)
  -> Nested Loop (cost=0.85..134449.89 rows=37 width=129) (actual time=3.784..81.096 rows=20 loops=1)
    Join Filter: (director.dni = d_persona.dni)
      -> Nested Loop (cost=0.43..134408.28 rows=37 width=92) (actual time=3.770..80.894 rows=20 loops=1)
        Join Filter: (director.sede_id = sede.id)
          -> Nested Loop (cost=0.14..134287.56 rows=37 width=73) (actual time=3.747..86.563 rows=20 loops=1)
            Join Filter: (director.sede_id = sede.id)
              Row Removed by Join Filter: 2086
              -> Index Scan using idx_sede_construccion_fecha on sede (cost=0.14..134172.06 rows=37 width=68) (actual time=3.701..80.288 rows=20 loops=1)
                Index Cond: ((construccion_fecha >= '1990-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date))
                Filter: (((SubPlan 1) + (SubPlan 2)) <= 408)
                Rows Removed by Filter: 1
                SubPlan 1
                  -> Aggregate (cost=165.12..165.12 rows=1 width=8) (actual time=0.535..0.535 rows=1 loops=21)
                    -> Seq Scan on profesor_sede (cost=0.00..165.00 rows=46 width=8) (actual time=0.011..0.529 rows=46 loops=21)
                      Filter: (sede_id = sede.id)
                      Row Removed by Filter: 9154
                SubPlan 2
                  -> Aggregate (cost=1032.62..1032.63 rows=1 width=8) (actual time=3.282..3.282 rows=1 loops=21)
                    -> Seq Scan on alumno (cost=0.00..1032.00 rows=250 width=0) (actual time=0.013..3.268 rows=233 loops=21)
                      Filter: (salon_sede_id = sede.id)
                      Row Removed by Filter: 49767
                  -> Materialize (cost=0.00..5.00 rows=200 width=13) (actual time=0.001..0.006 rows=185 loops=20)
                    -> Seq Scan on director (cost=0.00..4.00 rows=200 width=13) (actual time=0.009..0.019 rows=195 loops=1)
                    -> Index Scan using colaborador_pkey on colaborador d_colaborador (cost=0.29..3.26 rows=1 width=19) (actual time=0.013..0.013 rows=1 loops=20)
                      Index Cond: (dni = director.dni)
                    -> Index Scan using persona_pkey on persona d_persona (cost=0.42..1.08 rows=1 width=68) (actual time=0.008..0.008 rows=1 loops=20)
                      Index Cond: (dni = d_colaborador.dni)
Planning Time: 1.328 ms
Execution Time: 81.153 ms

```

- Para un millón de datos:



```

Limit  (cost=1.12..717133.06 rows=20 width=131) (actual time=197.745..631.710 rows=20 loops=1)
  -> Nested Loop  (cost=1.12..13231085.43 rows=369 width=131) (actual time=21.928..455.874 rows=20 loops=1)
    Join Filter: (director.dni = d_persona.dni)
      -> Nested Loop  (cost=0.70..13230674.09 rows=369 width=93) (actual time=21.888..455.528 rows=20 loops=1)
        -> Nested Loop  (cost=0.28..13229441.26 rows=369 width=74) (actual time=21.856..454.855 rows=20 loops=1)
          Join Filter: (director.sede_id = sede.id)
          Row Removed by Join Filter: 20991
          -> Index Scan using idx_sede_construcion_fecha on sede  (cost=0.28..13218335.26 rows=369 width=69) (actual time=21.581..453.049 rows=20 loops=1)
            Index Cond: ((construccion_fecha >= '1990-01-01'::date) AND (construccion_fecha <= '2010-12-31'::date))
            Filter: (((SubPlan 1) + (SubPlan 2)) <= 400)
            Rows Removed by Filter: 1
            SubPlan 1
              -> Aggregate  (cost=1612.11..1612.12 rows=1 width=8) (actual time=3.195..3.195 rows=1 loops=21)
                -> Seq Scan on profesor_sede  (cost=0.00..1612.00 rows=45 width=0) (actual time=0.053..3.186 rows=45 loops=21)
                  Filter: (sede_id = sede.id)
                  Rows Removed by Filter: 89955
            SubPlan 2
              -> Aggregate  (cost=10317.62..10317.64 rows=1 width=8) (actual time=18.364..18.364 rows=1 loops=21)
                -> Seq Scan on alumno  (cost=0.00..10317.00 rows=250 width=0) (actual time=0.075..18.335 rows=245 loops=21)
                  Filter: (salon_sede_id = sede.id)
                  Row Removed by Filter: 499755
              -> Materialize  (cost=0.00..41.00 rows=2000 width=13) (actual time=0.001..0.043 rows=1051 loops=20)
                -> Seq Scan on director  (cost=0.00..31.00 rows=2000 width=13) (actual time=0.007..0.089 rows=1888 loops=1)
              -> Index Scan using colaborador_pkey on colaborador d_colaborador  (cost=0.42..3.34 rows=1 width=19) (actual time=0.028..0.028 rows=1 loops=20)
                Index Cond: (dni = director.dni)
              -> Index Scan using persona_pkey on persona d_persona  (cost=0.42..1.09 rows=1 width=61) (actual time=0.014..0.014 rows=1 loops=20)
                Index Cond: (dni = d_colaborador.dni)
Planning Time: 0.862 ms
JIT:
  Functions: 35
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.179 ms, Inlining 9.038 ms, Optimization 101.788 ms, Emission 65.037 ms, Total 177.041 ms
Execution Time: 633.012 ms

```

- Descripción de las ejecuciones

- **Sin índices:** Los planes de consulta para las tablas de 1k y 1M son similares entre sí, con una operación extra de Gather en el millón de datos, mientras que los planes de 10k y 100k son similares pero difieren en comparación con los de 1k y 1M. Para 1k datos, se inicia con un Nested Loop Inner Join entre director y colaborador, seguido de otro Nested Loop Inner Join con persona y luego con sede. Las filas de sede se filtran usando un Seq Scan basado en sede.construccion\_fecha, y se ejecutan subconsultas “SubQuery Scan” para contar las filas de profesor\_sede y alumno, limitando el resultado a aquellos donde la suma no excede de 400. Estos resultados se ordenan “Sort” por sede.construccion\_fecha y se limita la salida a 20 filas. Para 10k datos, se cambia el orden, comenzando con un Nested Loop Inner Join entre director y colaborador, seguido de un Seq Scan sobre sede y un Nested Loop Inner Join con persona, aplicando las mismas subconsultas y limitando los resultados de manera similar. En 100k datos, el plan es similar al de 10k, pero se utiliza Parallel Seq Scan para manejar el mayor volumen de datos eficientemente, manteniendo las subconsultas y la ordenación de resultados. En el millón de datos, se observan los mismos pasos que en 100k, pero con el uso adicional de Parallel Seq Scan y Gather después del join para consolidar los resultados de las operaciones paralelas. En todos los casos, los planes dependen de Nested Loop Inner Join y Seq Scan, pero se aprovechan de operaciones paralelas a medida que el tamaño de los datos aumenta.
- **Con índices por defecto:** Los planes de consulta para las tablas de 1k, 10k, 100k y 1M datos muestran diferencias significativas en los algoritmos utilizados. Para 1k datos, el plan comienza con un Seq Scan en sede, filtrando por construccion\_fecha y limitando con subconsultas que cuentan las filas de profesor\_sede y alumno. Luego, se realiza un Nested Loop con director y colaborador, seguido de un Index Scan en persona, usando quicksort para ordenar. En 10k datos, el plan también inicia con Seq Scan en sede, pero las subconsultas se ejecutan más frecuentemente, con múltiples Nested Loop y Index Scan, manteniendo quicksort para la ordenación. Para 100k datos, se introduce un Materialize y top-N heapsort para ordenar, utilizando intensivamente Nested Loop y Seq Scan, aumentando el costo por el mayor volumen de datos. En 1M datos, se emplea Index Scan en sede.construccion\_fecha, mejorando la eficiencia. Se usan Nested Loop y Materialize para manejar los datos masivos, con top-N heapsort para ordenar y un uso efectivo de índices. Las subconsultas siguen siendo esenciales para los filtros en todos los casos.
- **Con índices por defecto más índices personalizados:** Los planes de consulta para tablas de 1k, 10k, 100k y 1M datos muestran optimizaciones significativas. Para 1k datos, el plan comienza con un Index Scan en sede utilizando el nuevo índice, seguido de un Seq Scan en profesor\_sede y alumno para las subconsultas de conteo, realizando después un Nested Loop con director y colaborador, y finalizando con un Index Scan en persona. Para 10k datos, el

plan también usa Index Scan en sede, pero las subconsultas se ejecutan más veces, resultando en múltiples Nested Loop y Index Scan, mejorando la eficiencia comparada con la versión sin índice. En 100k datos, se utiliza Index Scan en sede, y el Materialize se introduce para manejar mejor los datos, con top-N heapsort para la ordenación. Finalmente, para 1M datos, se observa un uso intensivo del Index Scan en sede, con múltiples Nested Loop, Materialize y Index Scan en otras tablas, optimizando el rendimiento significativamente. En todos los casos, las subconsultas y el uso del índice nuevo mejoran notablemente la eficiencia de la consulta.

### 5.3.2 Planes de índices para la segunda consulta

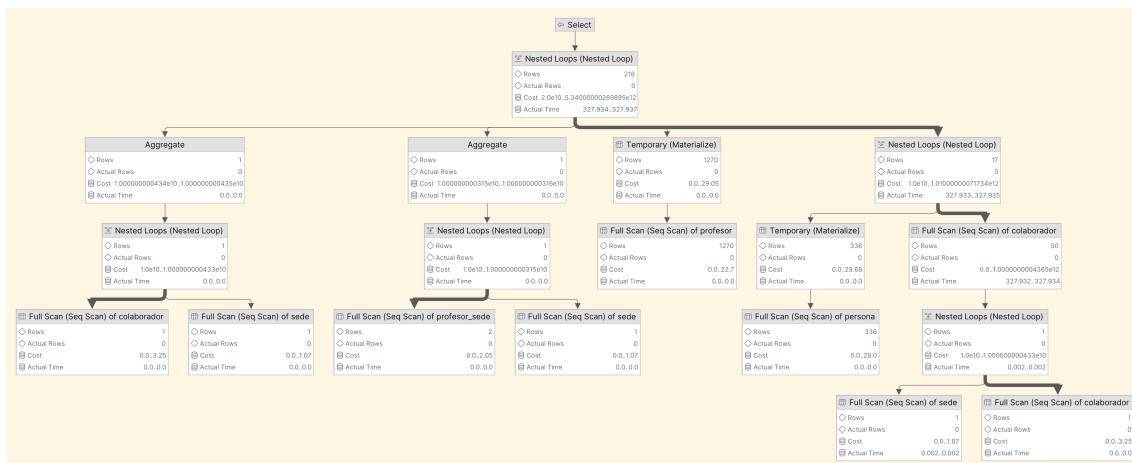
- Ejecución sin índices

```

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:

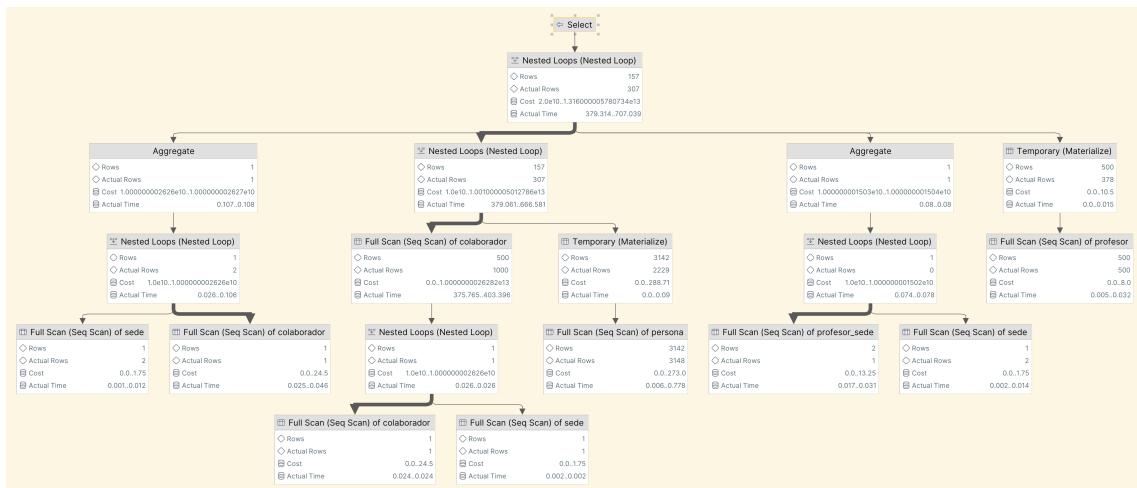


```

Nested Loop Left Join (cost=20000000000.00..5340000002698.95 rows=216 width=83) (actual time=305.255..305.257 rows=0 loops=1)
  Join Filter: (colaborador.dni = profesor.dni)
    -> Nested Loop (cost=10000000000.00..1010000000717.34 rows=17 width=92) (actual time=305.254..305.255 rows=0 loops=1)
      Join Filter: (colaborador.dni = persona.dni)
        -> Seq Scan on colaborador (cost=0.00..1000000000436.50 rows=50 width=42) (actual time=305.254..305.255 rows=0 loops=1)
          Filter: (esta.activo AND (SubPlan 3))
          Rows Removed By Filter: 100
        SubPlan 3
          -> Nested Loop (cost=10000000000.00..10000000004.33 rows=1 width=0) (actual time=0.001..0.001 rows=0 loops=100)
            -> Seq Scan on sede sede_2 (cost=0.00..1.07 rows=1 width=0) (actual time=0.001..0.001 rows=0 loops=100)
              Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FF
              Rows Removed By Filter: 2
            -> Seq Scan on colaborador_c_1 (cost=0.00..3.25 rows=1 width=0) (never executed)
              Filter: (dni = colaborador.dni)
            -> Materialize (cost=0.00..29.68 rows=336 width=59) (never executed)
              -> Seq Scan on persona (cost=0.00..28.00 rows=336 width=59) (never executed)
                Filter: ((nacimiento_fecha >= '1960-01-01'::date) AND (nacimiento_fecha <= '1980-12-31'::date))
            -> Materialize (cost=0.00..29.05 rows=1270 width=36) (never executed)
              -> Seq Scan on profesor (cost=0.00..22.70 rows=1270 width=36) (never executed)
        SubPlan 1
          -> Aggregate (cost=10000000003.15..10000000003.16 rows=1 width=8) (never executed)
            -> Nested Loop (cost=10000000000.00..10000000003.15 rows=1 width=4) (never executed)
              Join Filter: (sede.id = profesor_sede.sede_id)
                -> Seq Scan on sede (cost=0.00..1.07 rows=1 width=4) (never executed)
                  Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
                -> Seq Scan on profesor_sede (cost=0.00..2.05 rows=2 width=4) (never executed)
                  Filter: (profesor_dni = profesor.dni)
        SubPlan 2
          -> Aggregate (cost=10000000004.34..10000000004.35 rows=1 width=4) (never executed)
            -> Nested Loop (cost=10000000000.00..10000000004.33 rows=1 width=4) (never executed)
              -> Seq Scan on sede sede_1 (cost=0.00..1.07 rows=1 width=4) (never executed)
                Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
              -> Seq Scan on colaborador_c (cost=0.00..3.25 rows=1 width=0) (never executed)
                Filter: (dni = colaborador.dni)
Planning Time: 0.559 ms
JIT:
  Functions: 45
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 3.748 ms, Inlining 31.726 ms, Optimization 152.821 ms, Emission 120.408 ms, Total 308.703 ms
Execution Time: 309.166 ms

```

– Para diez mil datos:

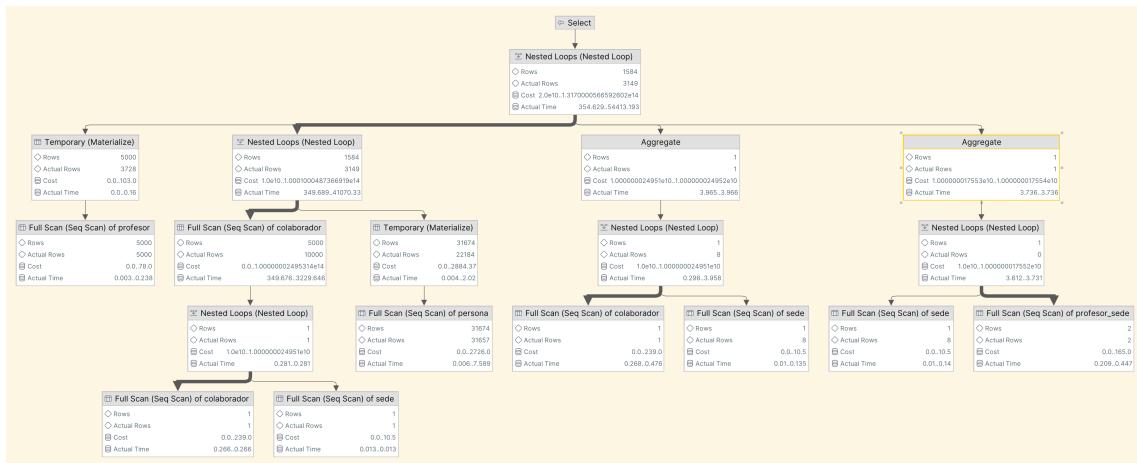


```

Nested Loop Left Join (cost=20000000000.00..1316000057807.34 rows=157 width=83) (actual time=395.100..707.557 rows=307 loops=1)
  Join Filter: (colaborador.dni = profesor.dni)
  Rows Removed by Join Filter: 115843
    > Nested Loop (cost=10000000000.00..10010000050127.86 rows=157 width=92) (actual time=394.869..668.077 rows=307 loops=1)
      Join Filter: (colaborador.dni = persona.dni)
      Rows Removed by Join Filter: 222853
        > Seq Scan on colaborador (cost=0.00..10000000026282.00 rows=500 width=42) (actual time=391.820..420.063 rows=1000 loops=1)
          Filter: (esta.activo AND (SubPlan 3))
          SubPlan 3
            > Nested Loop (cost=10000000000.00..10000000026.26 rows=1 width=0) (actual time=0.027..0.027 rows=1 loops=1000)
              > Seq Scan on sede sede_2 (cost=0.00..1.75 rows=1 width=0) (actual time=0.002..0.002 rows=1 loops=1000)
                Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1960-01-01'::date) AND ((nacimiento_fecha >='1960-01-01'::date) AND (nacimiento_fecha <='1980-12-31'::date)))
              Rows Removed by Filter: 500
            Rows Removed by Filter: 500
        > Materialize (cost=0.00..288.71 rows=3142 width=59) (actual time=0.000..0.005 rows=2229 loops=1000)
          > Seq Scan on persona (cost=0.00..273.00 rows=3142 width=59) (actual time=0.012..0.700 rows=3148 loops=1)
            Filter: ((nacimiento_fecha >='1960-01-01'::date) AND ((nacimiento_fecha <='1980-12-31'::date)))
            Rows Removed by Filter: 6852
      > Materialize (cost=0.00..500 width=9) (actual time=0.000..0.015 rows=378 loops=307)
        > Seq Scan on profesor (cost=0.00..8.00 rows=500 width=9) (actual time=0.007..0.031 rows=500 loops=1)
    SubPlan 1
      > Aggregate (cost=10000000015.03..10000000015.04 rows=1 width=8) (actual time=0.077..0.077 rows=1 loops=155)
        > Nested Loop (cost=10000000000.00..10000000015.02 rows=1 width=4) (actual time=0.073..0.076 rows=0 loops=155)
          Join Filter: (sede.id = profesor_sede.sede_id)
          Rows Removed by Join Filter: 3
          > Seq Scan on sede (cost=0.00..1.75 rows=1 width=4) (actual time=0.002..0.013 rows=2 loops=155)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1960-01-01'::date) AND ((nacimiento_fecha >='1960-01-01'::date) AND (nacimiento_fecha <='1980-12-31'::date)))
          Rows Removed by Filter: 18
          > Seq Scan on profesor_sede (cost=0.00..13.25 rows=2 width=4) (actual time=0.017..0.031 rows=1 loops=310)
            Filter: (profesor_dni = profesor.dni)
          Rows Removed by Filter: 714
    SubPlan 2
      > Aggregate (cost=10000000026.26..10000000026.27 rows=1 width=4) (actual time=0.104..0.104 rows=1 loops=152)
        > Nested Loop (cost=10000000000.00..10000000026.26 rows=1 width=4) (actual time=0.025..0.102 rows=2 loops=152)
          > Seq Scan on sede sede_1 (cost=0.00..1.75 rows=1 width=4) (actual time=0.002..0.012 rows=2 loops=152)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1960-01-01'::date) AND ((nacimiento_fecha >='1960-01-01'::date) AND (nacimiento_fecha <='1980-12-31'::date)))
          Rows Removed by Filter: 18
          > Seq Scan on colaborador c (cost=0.00..24.50 rows=1 width=0) (actual time=0.023..0.044 rows=1 loops=304)
            Filter: (dni = colaborador.dni)
          Rows Removed by Filter: 999
Planning Time: 0.562 ms
JIT:
  Functions: 45
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 2.267 ms, Inlining 39.979 ms, Optimization 214.747 ms, Emission 137.029 ms, Total 394.023 ms
Execution Time: 710.037 ms

```

– Para cien mil datos:



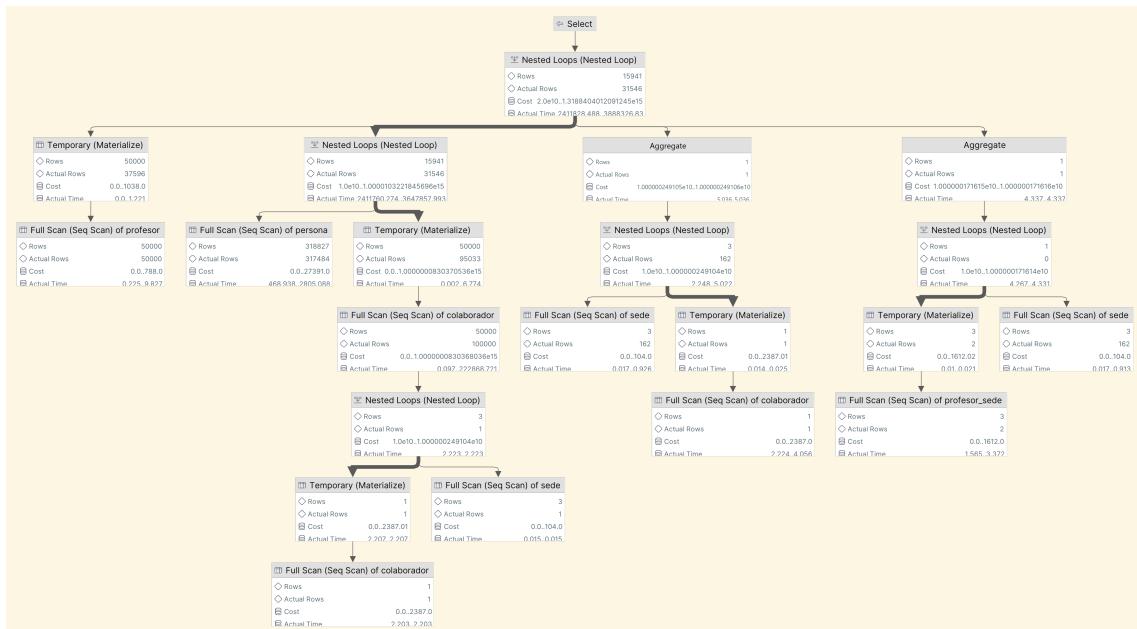
```

Nested Loop Left Join (cost=208000000000.00 .. 131708000566592.00 rows=1584 width=84) (actual time=389.204 .. 47239.921 rows=3149 loops=1)
  Join Filter: (colaborador.dni = profesor.dni)
  Rows Removed by Join Filter: 11738400
-> Nested Loop (cost=1000180004873669.19 rows=1584 width=93) (actual time=384.904 .. 35575.815 rows=3149 loops=1)
  Join Filter: (colaborador.dni = persona.dni)
  Rows Removed by Join Filter: 22183863
-> Seq Scan on colaborador (cost=0.00 .. 100000002495314.00 rows=5000 width=42) (actual time=384.892 .. 3049.835 rows=10000 loops=1)
  Filter: (esta_activo AND (SubPlan 3))
  SubPlan 3
    -> Nested Loop (cost=10000000000.00 .. 10000000249.51 rows=1 width=0) (actual time=0.259 .. 0.259 rows=1 loops=10000)
      -> Seq Scan on sede sede_2 (cost=0.00 .. 10.50 rows=1 width=0) (actual time=0.012 .. 0.012 rows=1 loops=10000)
        Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE) - 1) <='0':numeric) AND ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE) - 2) <='0':numeric)))
        Rows Removed by Filter: 7
      -> Seq Scan on colaborador c_1 (cost=0.00 .. 239.00 rows=1 width=0) (actual time=0.245 .. 0.245 rows=1 loops=10000)
        Filter: (dni = colaborador.dni)
        Rows Removed by Filter: 5000
-> Materialize (cost=0.00 .. 2084.37 rows=31674 width=60) (actual time=0.003 .. 1.753 rows=22184 loops=10000)
  -> Seq Scan on persona (cost=0.00 .. 2726.00 rows=31674 width=60) (actual time=0.006 .. 6.036 rows=31657 loops=1)
    Filter: ((nacimiento_fecha >= '1960-01-01':date) AND (nacimiento_fecha <= '1988-12-31':date))
  Rows Removed by Filter: 68343
-> Materialize (cost=0.00 .. 103.00 rows=5000 width=9) (actual time=0.001 .. 0.138 rows=3728 loops=3149)
  -> Seq Scan on profesor (cost=0.00 .. 78.00 rows=5000 width=9) (actual time=0.003 .. 0.227 rows=5000 loops=1)
SubPlan 1
-> Aggregate (cost=10000000175.53 .. 10000000175.54 rows=1 width=8) (actual time=3.173 .. 3.174 rows=1 loops=1579)
  -> Nested Loop (cost=10000000000.00 .. 10000000175.52 rows=1 width=4) (actual time=3.067 .. 3.169 rows=0 loops=1579)
    Join Filter: (sede.id = profesor.sede.sede_id)
    Rows Removed by Join Filter: 15
    -> Seq Scan on sede (cost=0.00 .. 10.50 rows=1 width=4) (actual time=0.008 .. 0.118 rows=8 loops=1579)
      Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE) - 1) <='0':numeric) AND ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE) - 2) <='0':numeric)))
      Rows Removed by Filter: 192
  -> Seq Scan on profesor_sede (cost=0.00 .. 165.00 rows=2 width=4) (actual time=0.176 .. 0.380 rows=2 loops=12632)
    Filter: (profesor.dni = profesor.dni)
    Rows Removed by Filter: 9157
SubPlan 2
-> Aggregate (cost=10000000249.51 .. 10000000249.52 rows=1 width=4) (actual time=3.555 .. 3.556 rows=1 loops=1570)
  -> Nested Loop (cost=10000000000.00 .. 10000000249.51 rows=1 width=4) (actual time=0.262 .. 3.548 rows=8 loops=1570)
    -> Seq Scan on sede sede_1 (cost=0.00 .. 10.50 rows=1 width=4) (actual time=0.008 .. 0.117 rows=8 loops=1570)
      Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE) - 1) <='0':numeric) AND ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE) - 2) <='0':numeric)))
      Rows Removed by Filter: 192
  -> Seq Scan on colaborador c (cost=0.00 .. 239.00 rows=1 width=0) (actual time=0.237 .. 0.428 rows=1 loops=12560)
    Filter: (dni = colaborador.dni)
    Rows Removed by Filter: 9999

Planning Time: 0.521 ms
JIT:
  Functions: 45
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 2.829 ms, Inlining 46.358 ms, Optimization 197.602 ms, Emission 140.962 ms, Total 387.751 ms
Execution Time: 47245.607 ms

```

– Para un millón de datos:



```

Nested Loop Left Join  (cost=20000000000.00..1318840401209124.50 rows=15941 width=85) (actual time=2408523.595..3931189.534 rows=31546 loops=1)
  Join Filter: (colaborador.dni = profesor.dni)
  Rows Removed by Join Filter: 1186002840
    -> Nested Loop  (cost=10000000000.00..1000010322184569.62 rows=15941 width=94) (actual time=2408470.840..3677830.753 rows=31546 loops=1)
      Join Filter: (colaborador.dni = persona.dni)
      Rows Removed by Join Filter: 3017152217
        -> Seq Scan on persona  (cost=0.00..27391.00 rows=318827 width=61) (actual time=493.714..2269.217 rows=317484 loops=1)
          Filter: ((nacimiento_fecha >= '1960-01-01'::date) AND (nacimiento_fecha <= '1980-12-31'::date))
          Rows Removed by Filter: 68251
        -> Materialize  (cost=0.00..1000000083037853.62 rows=50000 width=42) (actual time=0.002..6.796 rows=95033 loops=317484)
          Filter: (esta_activo AND (SubPlan 3))
          SubPlan 3
            -> Nested Loop  (cost=10000000000.00..10000002491.04 rows=3 width=0) (actual time=2.241..2.241 rows=1 loops=1000000)
              -> Seq Scan on colaborador  (cost=0.00..1000000083036803.62 rows=50000 width=42) (actual time=0.079..224709.902 rows=100000 loops=1)
                Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE)) - EXTRACT(year FROM construccion_fecha)) <= '10'::numeric))
                Rows Removed by Filter: 20
              -> Materialize  (cost=0.00..2387.01 rows=1 width=0) (actual time=2.22..2.224 rows=1 loops=100000)
                -> Seq Scan on profesor  (cost=0.00..2387.00 rows=1 width=0) (actual time=2.221..2.221 rows=1 loops=100000)
                  Filter: (dni = colaborador.dni)
                  Rows Removed by Filter: 50000
            -> Materialize  (cost=0.00..1038.00 rows=50000 width=0) (actual time=0.001..1.319 rows=37596 loops=31546)
              -> Seq Scan on profesor  (cost=0.00..788.00 rows=50000 width=0) (actual time=1.516..17.690 rows=50000 loops=1)
        SubPlan 1
          -> Aggregate  (cost=10000001716.15..10000001716.16 rows=1 width=0) (actual time=4.348..4.348 rows=1 loops=15696)
            -> Nested Loop  (cost=10000001716.00..10000001716.14 rows=1 width=4) (actual time=4.277..4.341 rows=0 loops=15696)
              Join Filter: (sede_id = profesor_sede.sede_id)
              Rows Removed by Join Filter: 294
              -> Seq Scan on sede  (cost=0.00..104.00 rows=3 width=4) (actual time=0.018..0.918 rows=162 loops=15696)
                Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) - EXTRACT(year FROM CURRENT_DATE)) <= '10'::numeric))
                Rows Removed by Filter: 1838
              -> Materialize  (cost=0.00..1612.02 rows=3 width=4) (actual time=0.010..0.021 rows=2 loops=2542752)
                -> Seq Scan on profesor_sede  (cost=0.00..1612.00 rows=3 width=4) (actual time=1.565..3.378 rows=2 loops=15696)
                  Filter: (profesor_dni = profesor.dni)
                  Rows Removed by Filter: 89998
          Planning Time: 5.218 ms
          JIT:
            Functions: 49
            Options: Inlining true, Optimization true, Expressions true, Deforming true
            Timing: Generation 2.500 ms, Inlining 127.223 ms, Optimization 220.763 ms, Emission 148.455 ms, Total 498.941 ms
          Timing: Generation 2.500 ms, Inlining 127.223 ms, Optimization 220.763 ms, Emission 148.455 ms, Total 498.941 ms
          Execution Time: 3931597.627 ms

```

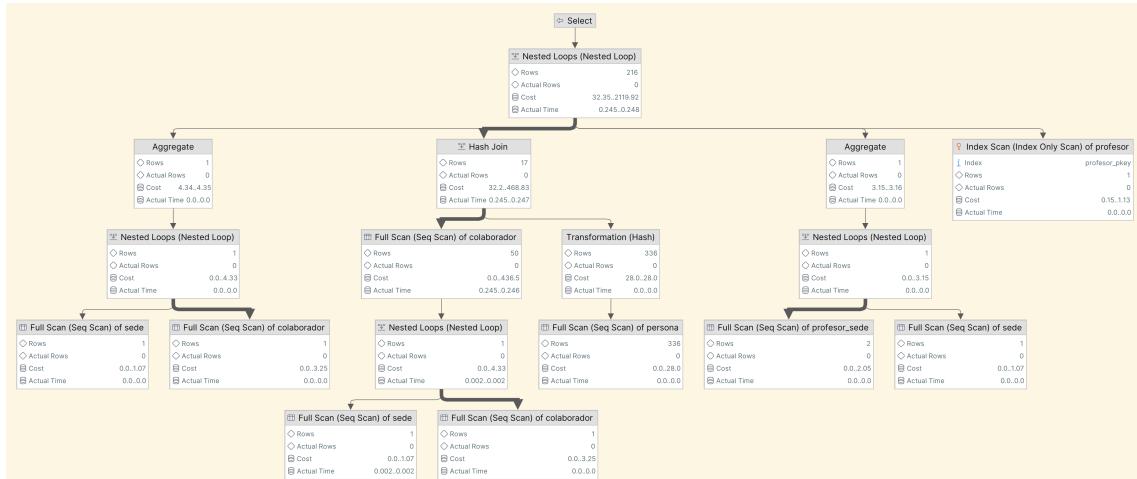
## • Ejecución con índices por defecto

```

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:

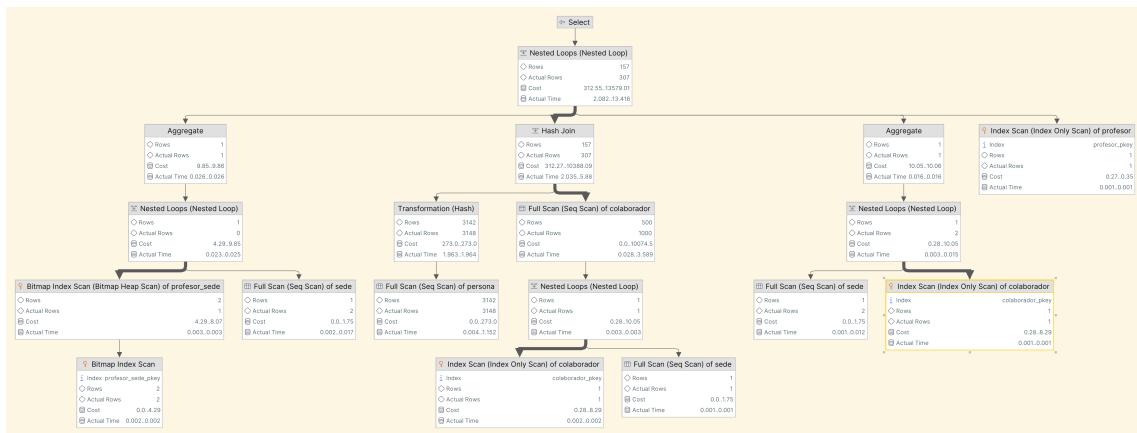


```

Nested Loop Left Join (cost=32.35..2119.92 rows=216 width=83) (actual time=0.369..0.372 rows=0 loops=1)
  -> Hash Join (cost=32.29..468.83 rows=17 width=92) (actual time=0.368..0.370 rows=0 loops=1)
    Hash Cond: (colaborador.dni = persona.dni)
    -> Seq Scan on colaborador (cost=0.00..436.50 rows=50 width=42) (actual time=0.368..0.369 rows=0 loops=1)
      Filter: (esta_activo AND (SubPlan 3))
      Rows Removed by Filter: 100
      SubPlan 3
        -> Nested Loop (cost=0.00..4.33 rows=1 width=0) (actual time=0.003..0.003 rows=0 loops=100)
          -> Seq Scan on sede sede_2 (cost=0.00..1.07 rows=1 width=0) (actual time=0.003..0.003 rows=0 loops=100)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FF
            Rows Removed by Filter: 2
            -> Seq Scan on colaborador c_1 (cost=0.00..3.25 rows=1 width=0) (never executed)
              Filter: (dni = colaborador.dni)
        -> Hash (cost=28.00..28.00 rows=336 width=59) (never executed)
          -> Seq Scan on persona (cost=0.00..28.00 rows=336 width=59) (never executed)
            Filter: ((nacimiento_fecha >= '1960-01-01':date) AND (nacimiento_fecha <= '1988-12-31':date))
    -> Index Only Scan using profesor_pkey on profesor (cost=0.15..1.13 rows=1 width=36) (never executed)
      Index Cond: (dni = colaborador.dni)
      Heap Fetches: 0
    SubPlan 1
      -> Aggregate (cost=3.15..3.16 rows=1 width=8) (never executed)
        -> Nested Loop (cost=0.00..3.15 rows=1 width=4) (never executed)
          Join Filter: (sede.id = profesor_sede.sede_id)
          -> Seq Scan on sede (cost=0.00..1.07 rows=1 width=4) (never executed)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
            -> Seq Scan on profesor_sede (cost=0.00..2.05 rows=2 width=4) (never executed)
              Filter: (profesor_dni = profesor.dni)
    SubPlan 2
      -> Aggregate (cost=4.34..4.35 rows=1 width=4) (never executed)
        -> Nested Loop (cost=0.00..4.33 rows=1 width=4) (never executed)
          -> Seq Scan on sede sede_1 (cost=0.00..1.07 rows=1 width=4) (never executed)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
            -> Seq Scan on colaborador c (cost=0.00..3.25 rows=1 width=0) (never executed)
              Filter: (dni = colaborador.dni)
Planning Time: 1.041 ms
Execution Time: 0.483 ms

```

– Para diez mil datos:

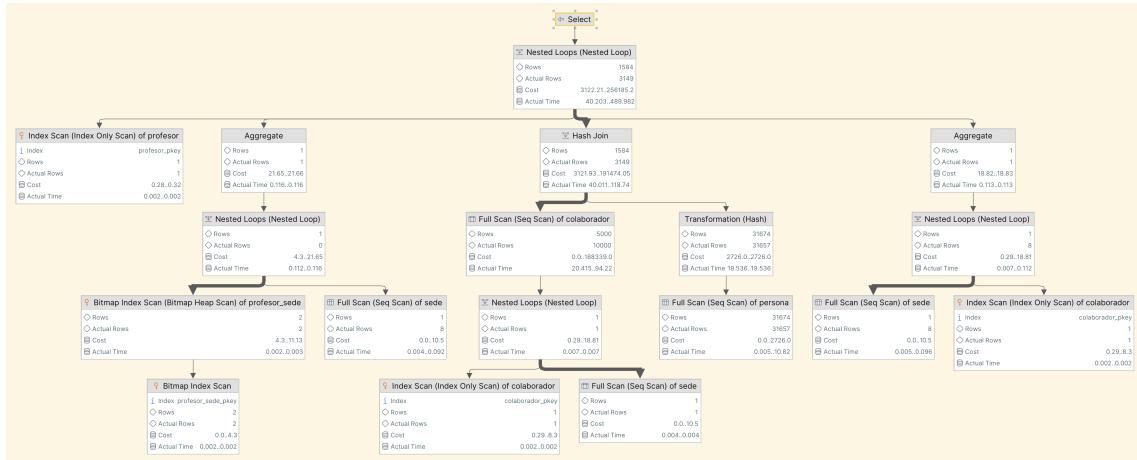


```

Nested Loop Left Join (cost=312.55..13579.01 rows=157 width=83) (actual time=1.721..19.317 rows=307 loops=1)
  -> Hash Join (cost=312.27..10388.09 rows=157 width=92) (actual time=1.685..7.154 rows=307 loops=1)
    Hash Cond: (colaborador.dni = persona.dni)
    -> Seq Scan on colaborador (cost=0.00..10074.50 rows=500 width=42) (actual time=0.019..5.100 rows=1000 loops=1)
      Filter: (esta_activo AND (SubPlan 3))
      SubPlan 3
        -> Nested Loop (cost=0.28..10.05 rows=1 width=0) (actual time=0.004..0.004 rows=1 loops=1000)
          -> Seq Scan on sede sede_2 (cost=0.00..1.75 rows=1 width=0) (actual time=0.001..0.001 rows=1 loops=1000)
            Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1968-01-01'::date) AND ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) <='1980-12-31'::date))
            Rows Removed by Filter: 6652
          Index Only Scan using colaborador_pkey on colaborador c_1 (cost=0.28..8.29 rows=1 width=0) (actual time=0.002..0.002 rows=1 loops=1000)
            Index Cond: (dni = colaborador.dni)
            Heap Fetches: 1000
        -> Hash (cost=273.00..273.00 rows=3142 width=59) (actual time=1.630..1.630 rows=3148 loops=1)
          Buckets: 4096 Batches: 1 Memory Usage: 316KB
          -> Seq Scan on persona (cost=0.00..273.00 rows=3142 width=59) (actual time=0.004..0.945 rows=3148 loops=1)
            Filter: ((nacimiento_fecha >= '1968-01-01'::date) AND (nacimiento_fecha <= '1980-12-31'::date))
            Rows Removed by Filter: 6652
      Index Only Scan using profesor_pkey on profesor (cost=0.27..0.35 rows=1 width=9) (actual time=0.002..0.002 rows=1 loops=307)
        Index Cond: (dni = colaborador.dni)
        Heap Fetches: 155
      SubPlan 1
        -> Aggregate (cost=9.85..9.86 rows=1 width=8) (actual time=0.030..0.030 rows=1 loops=155)
          -> Nested Loop (cost=4.29..9.85 rows=1 width=4) (actual time=0.027..0.029 rows=0 loops=155)
            Join Filter: (sede.id = profesor.sede.sede_id)
            Rows Removed by Join Filter: 3
            -> Seq Scan on sede sede_1 (cost=0.00..1.75 rows=1 width=4) (actual time=0.002..0.019 rows=2 loops=155)
              Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1968-01-01'::date) AND ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) <='1980-12-31'::date))
              Rows Removed by Filter: 18
            -> Bitmap Heap Scan on profesor_sede (cost=4.29..8.07 rows=2 width=4) (actual time=0.003..0.004 rows=1 loops=310)
              Recheck Cond: (profesor_dni = profesor.dni)
              Heap Blocks: exact=402
            -> Bitmap Index Scan on profesor_sede_pkey (cost=0.00..4.29 rows=2 width=0) (actual time=0.002..0.002 rows=2 loops=310)
              Index Cond: (profesor_dni = profesor.dni)
      SubPlan 2
        -> Aggregate (cost=10.05..10.06 rows=1 width=4) (actual time=0.040..0.040 rows=1 loops=152)
          -> Nested Loop (cost=0.28..10.05 rows=1 width=4) (actual time=0.006..0.039 rows=2 loops=152)
            -> Seq Scan on sede sede_1 (cost=0.00..1.75 rows=1 width=4) (actual time=0.001..0.033 rows=2 loops=152)
              Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1968-01-01'::date) AND ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) <='1980-12-31'::date))
              Rows Removed by Filter: 18
            -> Index Only Scan using colaborador_pkey on colaborador c (cost=0.28..8.29 rows=1 width=0) (actual time=0.003..0.003 rows=1 loops=304)
              Index Cond: (dni = colaborador.dni)
              Heap Fetches: 304
Planning Time: 0.622 ms
Execution Time: 19.401 ms

```

– Para cien mil datos:

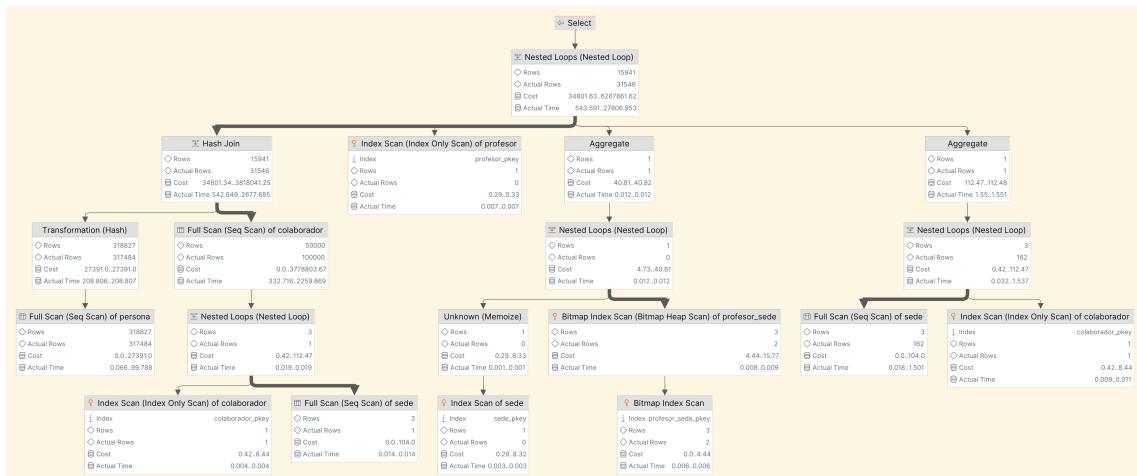


```

Nested Loop Left Join (cost=3122.21..256105.20 rows=1584 width=84) (actual time=67.101..674.827 rows=3149 loops=1)
  -> Hash Join (cost=3121.93..191474.05 rows=1584 width=93) (actual time=66.696..174.957 rows=3149 loops=1)
    Hash Cond: (colaborador.dni = persona.dni)
    -> Seq Scan on colaborador (cost=0.00..188339.00 rows=5000 width=42) (actual time=41.838..142.941 rows=10000 loops=1)
      Filter: (esta.activo AND (SubPlan 3))
      SubPlan 3
        -> Nested Loop (cost=0..29..18.81 rows=1 width=0) (actual time=0.009..0.009 rows=1 loops=10000)
          -> Seq Scan on sede sede_2 (cost=0.00..10.50 rows=1 width=0) (actual time=0.006..0.006 rows=1 loops=10000)
            Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM
              Rows Removed by Filter: 7
              -> Index Only Scan using colaborador_pkey on colaborador c_1 (cost=0.29..8.30 rows=1 width=0) (actual time=0.003..0.003 rows=1 loops=10000)
                Index Cond: (dni = colaborador.dni)
                Heap Fetches: 10000
              -> Hash (cost=2726.00..2726.00 rows=31674 width=60) (actual time=24.760..24.766 rows=31657 loops=1)
                Buckets: 32768 Batches: 1 Memory Usage: 3117KB
                -> Seq Scan on persona (cost=0.00..2726.00 rows=31674 width=60) (actual time=0.007..13.694 rows=31657 loops=1)
                  Filter: ((nacimiento_fecha >='1960-01-01':date) AND (nacimiento_fecha <='1980-12-31':date))
                  Rows Removed by Filter: 68343
                -> Index Only Scan using profesor_pkey on profesor (cost=0.28..0.32 rows=1 width=9) (actual time=0.003..0.003 rows=3149)
                  Index Cond: (dni = colaborador.dni)
                  Heap Fetches: 0
                SubPlan 1
                  -> Aggregate (cost=21.65..21.66 rows=1 width=8) (actual time=0.174..0.174 rows=1 loops=1579)
                    -> Nested Loop (cost=4.30..21.65 rows=1 width=4) (actual time=0.167..0.173 rows=0 loops=1579)
                      Join Filter: (sede.id = profesor.sede.sede_id)
                      Rows Removed by Join Filter: 15
                      -> Seq Scan on sede (cost=0.00..10.50 rows=1 width=4) (actual time=0.006..0.134 rows=8 loops=1579)
                        Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
                          Rows Removed by Filter: 192
                          -> Bitmap Heap Scan on profesor_sede (cost=4.30..11.13 rows=2 width=4) (actual time=0.003..0.004 rows=2 loops=12632)
                            Recheck Cond: (profesor.dni = profesor.dni)
                            Heap Blocks: exact-22584
                            -> Bitmap Index Scan on profesor_sede_pkey (cost=0.00..4.30 rows=2 width=0) (actual time=0.003..0.003 rows=2 loops=12632)
                              Index Cond: (profesor.dni = profesor.dni)
                              Heap Fetches: 12560
                SubPlan 2
                  -> Aggregate (cost=18.82..18.83 rows=1 width=4) (actual time=0.133..0.133 rows=1 loops=1570)
                    -> Nested Loop (cost=0.29..18.81 rows=1 width=4) (actual time=0.007..0.132 rows=8 loops=1570)
                      -> Seq Scan on sede sede_1 (cost=0.00..10.50 rows=1 width=4) (actual time=0.005..0.113 rows=8 loops=1570)
                        Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
                          Rows Removed by Filter: 192
                          -> Index Only Scan using colaborador_pkey on colaborador c (cost=0.29..8.30 rows=1 width=0) (actual time=0.002..0.002 rows=1 loops=12560)
                            Index Cond: (dni = colaborador.dni)
                            Heap Fetches: 12560
              Planning Time: 0.997 ms
              JIT:
                Functions: 47
                Options: Inlining false, Optimization false, Expressions true, Deforming true
                Timing: Generation 2.546 ms, Inlining 0.000 ms, Optimization 1.630 ms, Emission 40.224 ms, Total 44.400 ms
              Execution Time: 677.966 ms

```

– Para un millón de datos:



```

Nested Loop Left Join (cost=34801.63 .. 6267861.62 rows=15941 width=85) (actual time=492.034..18938.484 rows=31546 loops=1)
  -> Hash Join (cost=34801.34 .. 3818041.25 rows=15941 width=94) (actual time=491.167..1931.962 rows=31546 loops=1)
    Hash Cond: (colaborador.dni = persona.dni)
    -> Seq Scan on colaborador (cost=0.00 .. 3778883.67 rows=500000 width=42) (actual time=298.722..1618.964 rows=100000 loops=1)
      Filter: (esta_activo AND (subPlan 3))
      SubPlan 3
        -> Nested Loop (cost=0.42 .. 112.47 rows=3 width=0) (actual time=0.013..0.013 rows=1 loops=100000)
          -> Index Only Scan using colaborador_pkey on colaborador_c_1 (cost=0.42 .. 8.44 rows=1 width=0) (actual time=0.003..0.003 rows=1 loops=100000)
            Index Cond: (dni = colaborador.dni)
            Heap Fetches: 100000
          -> Seq Scan on sede_sede_2 (cost=0.00 .. 104.00 rows=3 width=0) (actual time=0.010 .. 0.010 rows=1 loops=100000)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FF
              Rows Removed by Filter: 28
-> Hash (cost=2.7391 .. 2.7391.00 rows=318827 width=61) (actual time=191.263..191.264 rows=317484 loops=1)
  Buckets: 131072 Batches: 8 Memory Usage: 4638K
  -> Seq Scan on persona (cost=0.00 .. 27391.00 rows=318827 width=61) (actual time=0.021 .. 83.912 rows=317484 loops=1)
    Filter: ((nacimiento_fecha >= '1960-01-01'::date) AND (nacimiento_fecha <= '1980-12-31'::date))
    Rows Removed by Filter: 682516
-> Index Only Scan using profesor_pkey on profesor (cost=0.29 .. 0.33 rows=1 width=9) (actual time=0.004 .. 0.004 rows=0 loops=31546)
  Index Cond: (dni = colaborador.dni)
  Heap Fetches: 0
SubPlan 1
  -> Aggregate (cost=40.81 .. 48.82 rows=1 width=8) (actual time=0.007 .. 0.007 rows=1 loops=15696)
    -> Nested Loop (cost=4.73 .. 48.81 rows=1 width=4) (actual time=0.006 .. 0.006 rows=0 loops=15696)
      -> Bitmap Heap Scan on profesor_sede (cost=4.44 .. 15.77 rows=3 width=4) (actual time=0.004 .. 0.005 rows=2 loops=15696)
        Recheck Cond: (profesor.dni = profesor.sede)
        Heap Blocks: exact=28450
      -> Bitmap Index Scan on profesor_sede_pkey (cost=0.00 .. 4.44 rows=3 width=0) (actual time=0.003 .. 0.003 rows=2 loops=15696)
        Index Cond: (profesor.dni = profesor.sede)
        Index Only Scan using profesor_sede_sede_id (cost=0.29 .. 8.33 rows=1 width=4) (actual time=0.000 .. 0.000 rows=0 loops=28497)
        Cache Key: profesor_sede.sede_id
        Cache Mode: logical
        Hits: 26497 Misses: 2000 Evictions: 0 Overflows: 0 Memory Usage: 139KB
      -> Index Scan using sede_pkey on sede (cost=0.28 .. 8.32 rows=1 width=4) (actual time=0.002 .. 0.002 rows=0 loops=2800)
        Index Cond: (id = profesor_sede.sede_id)
        Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FF
          Rows Removed by Filter: 1
-> SubPlan 2
  -> Aggregate (cost=12.47 .. 112.48 rows=1 width=4) (actual time=1.054 .. 1.054 rows=1 loops=15850)
    -> Nested Loop (cost=0.42 .. 112.47 rows=3 width=4) (actual time=0.019 .. 1.046 rows=162 loops=15850)
      -> Index Only Scan using colaborador_pkey on colaborador_c (cost=0.42 .. 8.44 rows=1 width=0) (actual time=0.005 .. 0.006 rows=1 loops=15850)
        Index Cond: (dni = colaborador.dni)
        Heap Fetches: 15850
      -> Seq Scan on sede_sede_1 (cost=0.00 .. 104.00 rows=3 width=4) (actual time=0.012 .. 1.025 rows=162 loops=15850)
        Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FF
          Rows Removed by Filter: 1838
Planning Time: 5.759 ms
JIT:
  Functions: 51
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 3.766 ms, Inlining 26.068 ms, Optimization 143.440 ms, Emission 129.215 ms, Total 302.488 ms
Execution Time: 18945.817 ms

```

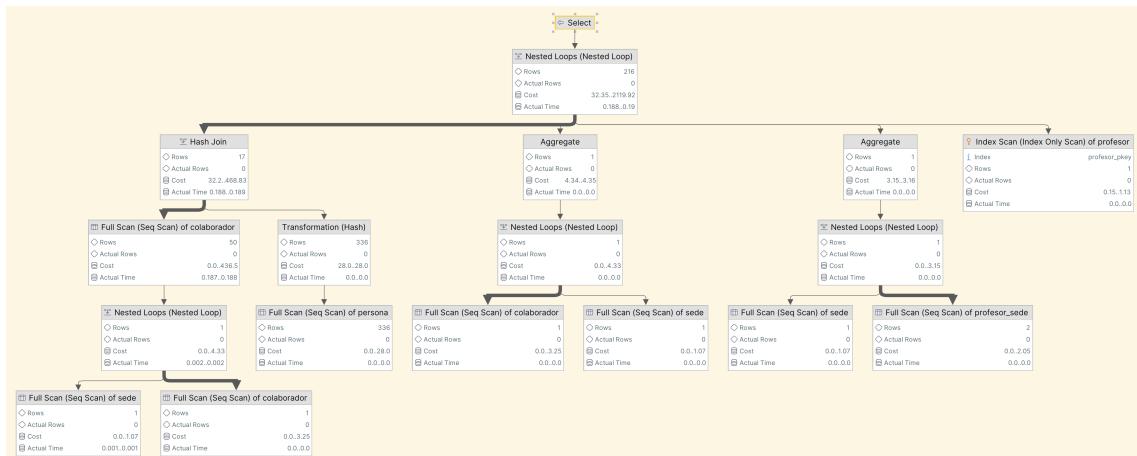
- Ejecución con índices por defecto más índices personalizados

```

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 VACUUM FULL colaborador;
10 VACUUM FULL persona;
11 VACUUM FULL profesor_sede;
12 VACUUM FULL sede;
13
14 CREATE INDEX IF NOT EXISTS idx_persona_nacimiento_fecha ON persona (
  nacimiento_fecha);

```

– Para mil datos:

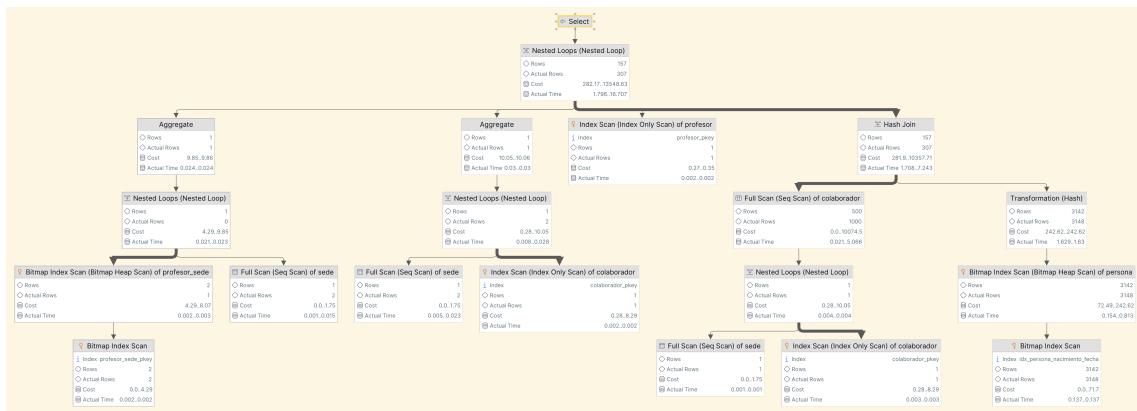


```

Nested Loop Left Join (cost=32.35..2119.92 rows=216 width=83) (actual time=0.178..0.180 rows=0 loops=1)
  -> Hash Join (cost=32.29..468.83 rows=17 width=92) (actual time=0.178..0.179 rows=0 loops=1)
    Hash Cond: (colaborador.dni = persona.dni)
    -> Seq Scan on colaborador (cost=0.00..436.50 rows=50 width=42) (actual time=0.178..0.178 rows=0 loops=1)
      Filter: (esta_activo AND (SubPlan 3))
      Rows Removed by Filter: 100
      SubPlan 3
        -> Nested Loop (cost=0.00..4.33 rows=1 width=0) (actual time=0.001..0.001 rows=0 loops=100)
          -> Seq Scan on sede sede_2 (cost=0.00..1.07 rows=1 width=0) (actual time=0.001..0.001 rows=0 loops=100)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM FF
              Rows Removed by Filter: 2
              -> Seq Scan on colaborador c_1 (cost=0.00..3.25 rows=1 width=0) (never executed)
                Filter: (dni = colaborador.dni)
              -> Hash (cost=28.00..28.00 rows=336 width=59) (never executed)
                -> Seq Scan on persona (cost=0.00..28.00 rows=336 width=59) (never executed)
                  Filter: ((nacimiento_fecha >= '1960-01-01':date) AND (nacimiento_fecha <= '1988-12-31':date))
              -> Index Only Scan using profesor_pkey on profesor (cost=0.15..1.13 rows=1 width=36) (never executed)
                Index Cond: (dni = colaborador.dni)
                Heap Fetches: 0
              SubPlan 1
                -> Aggregate (cost=3.15..3.16 rows=1 width=8) (never executed)
                  -> Nested Loop (cost=0.00..3.15 rows=1 width=4) (never executed)
                    Join Filter: (sede.id = profesor_sede.sede_id)
                    -> Seq Scan on sede (cost=0.00..1.07 rows=1 width=4) (never executed)
                      Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
                        -> Seq Scan on profesor_sede (cost=0.00..2.05 rows=2 width=4) (never executed)
                          Filter: (profesor_dni = profesor.dni)
              SubPlan 2
                -> Aggregate (cost=4.34..4.35 rows=1 width=4) (never executed)
                  -> Nested Loop (cost=0.00..4.33 rows=1 width=4) (never executed)
                    -> Seq Scan on sede sede_1 (cost=0.00..1.07 rows=1 width=4) (never executed)
                      Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
                        -> Seq Scan on colaborador c (cost=0.00..3.25 rows=1 width=0) (never executed)
                          Filter: (dni = colaborador.dni)
            Planning Time: 0.690 ms
            Execution Time: 0.224 ms

```

– Para diez mil datos:

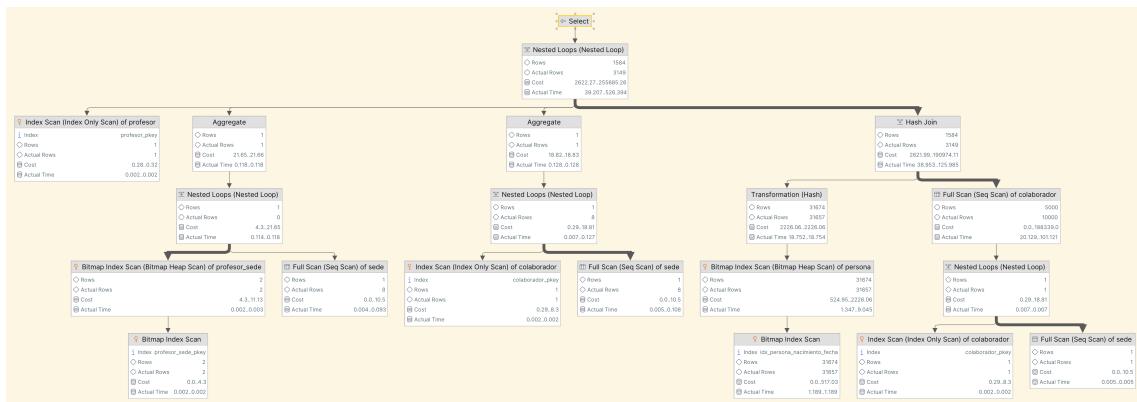


```

Nested Loop Left Join (cost=282.17..13548.63 rows=157 width=83) (actual time=1.575..14.364 rows=307 loops=1)
  -> Hash Join (cost=281.99..10357.71 rows=157 width=92) (actual time=1.530..5.907 rows=307 loops=1)
    Hash Cond: (colaborador.dni = persona.dni)
    -> Seq Scan on colaborador (cost=0..0.10074.50 rows=500 width=42) (actual time=0.023..4.009 rows=1000 loops=1)
      Filter: (esta_activo AND (SubPlan 3))
      SubPlan 3
        -> Nested Loop (cost=0..28..10.05 rows=1 width=0) (actual time=0.003..0.003 rows=1 loops=1000)
          -> Seq Scan on sede sede_2 (cost=0..0.175 rows=1 width=0) (actual time=0.001..0.001 rows=1 loops=1000)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1980-12-31':date) AND ((haciimiento_fecha >='1960-01-01':date) AND (haciimiento_fecha <='1980-12-31':date)))
            Index Cond: (dni = colaborador.dni)
            Heap Fetches: 1000
        -> Hash (cost=242.62..242.62 rows=3142 width=59) (actual time=1.472..1.473 rows=3148 loops=1)
          Buckets: 4096 Batches: 1 Memory Usage: 316KB
          -> Bitmap Heap Scan on persona (cost=72.49..242.62 rows=3142 width=59) (actual time=0.157..0.717 rows=3148 loops=1)
            Recheck Cond: ((haciimiento_fecha >='1960-01-01':date) AND (haciimiento_fecha <='1980-12-31':date))
            Heap Blocks: exact=123
            -> Bitmap Index Scan on idx_persona_nacimiento_fecha (cost=0..0.71 width=0) (actual time=0.140..0.140 rows=3148 loops=1)
              Index Cond: ((nacimiento_fecha >='1960-01-01':date) AND (nacimiento_fecha <='1980-12-31':date))
            Index Only Scan using profesor_pkey on profesor (cost=0.27..0.35 rows=1 width=9) (actual time=0.002..0.002 rows=1 loops=307)
              Index Cond: (dni = colaborador.dni)
              Heap Fetches: 155
            SubPlan 1
              -> Aggregate (cost=9.85..9.86 rows=1 width=8) (actual time=0.024..0.024 rows=1 loops=155)
                -> Nested Loop (cost=4.29..9.85 rows=1 width=4) (actual time=0.022..0.024 rows=0 loops=155)
                  Join Filter: (sede.id = profesor.sede_id)
                  Rows Removed by Join Filter: 3
                  -> Seq Scan on sede (cost=0..0.175 rows=1 width=4) (actual time=0.001..0.015 rows=2 loops=155)
                    Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1980-12-31':date) AND ((haciimiento_fecha >='1960-01-01':date) AND (haciimiento_fecha <='1980-12-31':date)))
                    Rows Removed by Filter: 18
                  -> Bitmap Heap Scan on profesor_sede (cost=4.29..8.07 rows=2 width=4) (actual time=0.003..0.004 rows=1 loops=310)
                    Recheck Cond: (profesor_dni = profesor.dni)
                    Heap Blocks: exact=402
                    -> Bitmap Index Scan on profesor_sede_pkey (cost=0..0.49 rows=2 width=0) (actual time=0.002..0.002 rows=2 loops=310)
                      Index Cond: (profesor_dni = profesor.dni)
            SubPlan 2
              -> Aggregate (cost=10.05..10.06 rows=1 width=4) (actual time=0.024..0.024 rows=1 loops=152)
                -> Nested Loop (cost=0..28..10.05 rows=1 width=4) (actual time=0.004..0.023 rows=2 loops=152)
                  -> Seq Scan on sede sede_1 (cost=0..0.175 rows=1 width=4) (actual time=0.001..0.018 rows=2 loops=152)
                    Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '1980-12-31':date) AND ((haciimiento_fecha >='1960-01-01':date) AND (haciimiento_fecha <='1980-12-31':date)))
                    Rows Removed by Filter: 18
                  -> Index Only Scan using colaborador_pkey on colaborador_c (cost=0..28..8.29 rows=1 width=0) (actual time=0.002..0.002 rows=1 loops=304)
                    Index Cond: (dni = colaborador.dni)
                    Heap Fetches: 304
            Planning Time: 4.055 ms
            Execution Time: 14.478 ms

```

– Para cien mil datos:

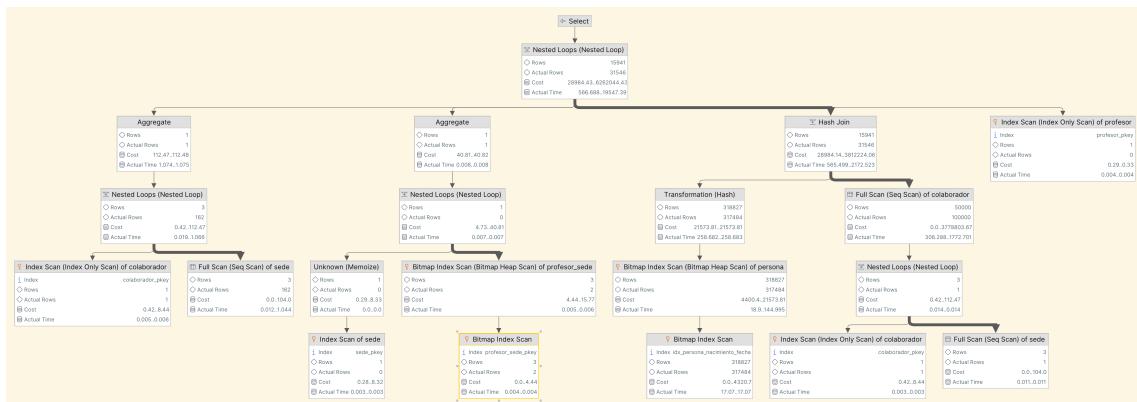


```

Nested Loop Left Join (cost=2622.27..255685.26 rows=1584 width=84) (actual time=52.767..559.760 rows=3149 loops=1)
  -> Hash Join (cost=2621.99..199974.11 rows=1584 width=93) (actual time=52.461..139.479 rows=3149 loops=1)
    Hash Cond: (colaborador.dni = persona.dni)
    -> Seq Scan on colaborador (cost=0.00..188339.00 rows=5000 width=42) (actual time=25.635..107.606 rows=10000 loops=1)
      Filter: (esta_activo AND (SubPlan 3))
      SubPlan 3
        -> Nested Loop (cost=0.29..18.81 rows=1 width=0) (actual time=0.088..0.088 rows=1 loops=10000)
          -> Seq Scan on sede sede_2 (cost=0.00..10.50 rows=1 width=0) (actual time=0.085..0.085 rows=1 loops=10000)
            Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FF
              Rows Removed by Filter: 7
              -> Index Only Scan using colaborador_pkey on colaborador_c_1 (cost=0.29..8.30 rows=1 width=0) (actual time=0.082..0.082 rows=1 loops=10000)
                Index Cond: (dni = colaborador.dni)
                Heap Fetches: 10800
              -> Hash (cost=2226.06..2226.06 rows=31674 width=60) (actual time=26.747..26.747 rows=31657 loops=1)
                Buckets: 32768 Batches: 1 Memory Usage: 317KB
                -> Bitmap Heap Scan on persona (cost=524.95..2226.06 rows=31674 width=60) (actual time=1.793..11.336 rows=31657 loops=1)
                  Recheck Cond: ((nacimiento_fecha >= '1960-01-01'::date) AND (nacimiento_fecha <= '1980-12-31'::date))
                  Heap Blocks: exact=1226
                  -> Bitmap Index Scan on idx_persona_nacimiento_fecha (cost=0.00..517.03 rows=31674 width=0) (actual time=1.610..1.610 rows=31657 loops=1)
                    Index Cond: ((nacimiento_fecha >= '1960-01-01'::date) AND (nacimiento_fecha <= '1980-12-31'::date))
                  -> Index Only Scan using profesor_pkey on profesor (cost=0.28..0.32 rows=1 width=9) (actual time=0.082..0.082 rows=1 loops=3149)
                    Index Cond: (dni = colaborador.dni)
                    Heap Fetches: 8
              SubPlan 1
                -> Aggregate (cost=21.65..21.66 rows=1 width=8) (actual time=0.136..0.136 rows=1 loops=1579)
                  -> Nested Loop (cost=4.30..21.65 rows=1 width=4) (actual time=0.130..0.135 rows=0 loops=1579)
                    Join Filter: (sede.id = profesor.sede.sede_id)
                    Rows Removed by Join Filter: 15
                    -> Seq Scan on sede (cost=0.00..10.50 rows=1 width=4) (actual time=0.085..0.105 rows=8 loops=1579)
                      Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
                        Rows Removed by Filter: 192
                        -> Bitmap Heap Scan on profesor_sede (cost=4.30..11.13 rows=2 width=4) (actual time=0.003..0.003 rows=2 loops=12632)
                          Recheck Cond: (profesor.dni = profesor.dni)
                          Heap Blocks: exact=22584
                          -> Bitmap Index Scan on profesor_sede_pkey (cost=0.00..4.30 rows=2 width=0) (actual time=0.002..0.002 rows=2 loops=12632)
                            Index Cond: (profesor_dni = profesor.dni)
              SubPlan 2
                -> Aggregate (cost=18.82..18.83 rows=1 width=4) (actual time=0.124..0.124 rows=1 loops=1570)
                  -> Nested Loop (cost=0.29..18.81 rows=1 width=4) (actual time=0.087..0.122 rows=8 loops=1570)
                    -> Seq Scan on sede sede_1 (cost=0.00..10.50 rows=1 width=4) (actual time=0.085..0.105 rows=8 loops=1570)
                      Filter: (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0':numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM cor
                        Rows Removed by Filter: 192
                        -> Index Only Scan using colaborador_pkey on colaborador_c (cost=0.29..8.30 rows=1 width=0) (actual time=0.082..0.082 rows=1 loops=12560)
                          Index Cond: (dni = colaborador.dni)
                          Heap Fetches: 12560
            Planning Time: 0.882 ms
            JIT:
              Functions: 47
              Options: Inlining false, Optimization false, Expressions true, Deforming true
              Timing: Generation 3.236 ms, Inlining 0.000 ms, Optimization 1.488 ms, Emission 24.244 ms, Total 28.969 ms
            Execution Time: 563.422 ms

```

– Para un millón de datos:



```

Nested Loop Left Join (cost=28984.43 . 6262844 . 43 rows=15941 width=85) (actual time=432.018 .. 16847.241 rows=31546 loops=1)
-> Hash Join (cost=28984.14 .. 381224.06 rows=15941 width=94) (actual time=432.212 .. 1925.587 rows=31546 loops=1)
  Hash Cond: (colaborador.dni = persona.dni)
    > Seq Scan on colaborador (cost=0.00 .. 3778803.67 rows=50000 width=42) (actual time=268.423 .. 1657.018 rows=100000 loops=1)
      Filter: (esta_activo AND (Subplan 3))
      SubPlan 3
        >- Nested Loop (cost=0.42 .. 112.47 rows=3 width=0) (actual time=0.014 .. 0.014 rows=1 loops=100000)
          Index Only Scan using colaborador_pkey on colaborador c_1 (cost=0.42 .. 8.44 rows=1 width=0) (actual time=0.003 .. 0.003 rows=1 loops=100000)
            Index Cond: (dni = colaborador.dni)
            Heap Fetches: 100000
          > Seq Scan on sede sede_2 (cost=0.00 .. 104.00 rows=3 width=0) (actual time=0.010 .. 6.010 rows=1 loops=100000)
            Index Cond: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE)) - EXTRACT(year FROM construccion_fecha)) < 100)
            Rows Removed by Filter: 20
-> Hash (cost=21573.81 .. 21573.81 rows=31827 width=61) (actual time=163.232 .. 163.233 rows=317484 loops=1)
  Buckets: 131072 Batches: 8 Memory Usage: 4630KB
    > Bitmap Heap Scan on persona (cost=4400.40 .. 21573.81 rows=31827 width=61) (actual time=12.922 .. 72.701 rows=317484 loops=1)
      Recheck Cond: ((nacimiento_fecha >='1960-01-01':date) AND (nacimiento.fecha <='1980-12-31':date))
      Heap Blocks: exact=12391
      > Bitmap Index Scan on idx_persona_nacimiento_fecha (cost=0.00 .. 4320.70 rows=318007 width=0) (actual time=11.639 .. 11.639 rows=317484 loops=1)
        Index Cond: ((nacimiento.fecha >='1960-01-01':date) AND (nacimiento.fecha <='1980-12-31':date))
-> Index Only Scan using profesor_pkey on profesor (cost=0.29 .. 0.33 rows=1 width=9) (actual time=0.003 .. 0.003 rows=0 loops=31546)
  Index Cond: (dni = colaborador.dni)
  Heap Fetches: 0
SubPlan 1
  >- Aggregate (cost=0.21 .. 40.82 rows=1 width=0) (actual time=0.006 .. 0.006 rows=1 loops=15696)
    >- Nested Loop (cost=0.73 .. 49.81 rows=1 width=0) (actual time=0.006 .. 0.006 rows=0 loops=15696)
      > Bitmap Heap Scan on profesor_sede (cost=1.44 .. 15.77 rows=3 width=4) (actual time=0.004 .. 0.005 rows=2 loops=15696)
        Recheck Cond: (profesor.dni = profesor.dni)
        Heap Blocks: exact=28450
        > Bitmap Index Scan on profesor_sede_pkey (cost=0.00 .. 4.44 rows=3 width=0) (actual time=0.003 .. 0.003 rows=2 loops=15696)
          Index Cond: (profesor.dni = profesor.dni)
          Heap Fetches: 1
      >- Memoize (cost=0.20 .. 8.33 rows=1 width=4) (actual time=0.000 .. 0.000 rows=0 loops=28497)
        Cache Key: profesor_sede.sede_id
        Cache Mode: logical
        Hints: 26497 Misses: 2000 Evictions: 0 Overflows: 0 Memory Usage: 139KB
        > Index Scan using sede_pkey on sede (cost=0.28 .. 28.82 rows=1 width=4) (actual time=0.003 .. 0.003 rows=0 loops=2000)
          Index Cond: (sede_id = profesor_sede.sede_id)
          Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) > '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE)) - EXTRACT(year FROM construccion_fecha)) < 100)
          Rows Removed by Filter: 1
SubPlan 2
  >- Aggregate (cost=112.47 .. 112.48 rows=1 width=4) (actual time=0.926 .. 0.926 rows=1 loops=15850)
    >- Nested Loop (cost=0.42 .. 112.47 rows=3 width=4) (actual time=0.016 .. 0.019 rows=162 loops=15850)
      > Index Only Scan using colaborador_pkey on colaborador c (cost=0.42 .. 8.44 rows=1 width=0) (actual time=0.004 .. 0.005 rows=1 loops=15850)
        Index Cond: (dni = colaborador.dni)
        Heap Fetches: 15850
    >- Seq Scan on sede sede_1 (cost=0.00 .. 104.00 rows=3 width=4) (actual time=0.010 .. 0.002 rows=162 loops=15850)
      Filter: ((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM construccion_fecha)) < '0'::numeric) AND (((EXTRACT(year FROM CURRENT_DATE) - EXTRACT(year FROM CURRENT_DATE)) - EXTRACT(year FROM construccion_fecha)) < 100)
      Rows Removed by Filter: 1838
Planning Time: 0.953 ms
JIT:
  Functions: 51
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 4.659 ms, Inlining 34.497 ms, Optimization 138.122 ms, Emission 95.787 ms, Total 273.065 ms
Execution Time: 16854.632 ms

```

- Descripción de las ejecuciones

- **Sin índices:** Los planes de consulta para las tablas de 1k, 10k, 100k y 1M datos muestran diferencias significativas en la ejecución. Para 1k datos, el plan utiliza un Nested Loop Left Join entre colaborador y profesor, con un Seq Scan en colaborador y subconsultas que filtran por sede. Se realiza una materialización y un Seq Scan en persona para aplicar el filtro de nacimiento\_fecha, y un Seq Scan en profesor. Para 10k datos, se repite el uso de Nested Loop Left Join y Seq Scan en colaborador, pero con más iteraciones y materializaciones, incrementando los costos de la consulta. En 100k datos, el plan sigue siendo similar, pero se observa un aumento significativo en la cantidad de subconsultas y materializaciones necesarias. Finalmente, para 1M datos, el plan utiliza Seq Scan en persona y colaborador con múltiples Nested Loop Left Join y subconsultas, resultando en una mayor complejidad en la ejecución. En todos los casos, los Seq Scan y Nested Loop predominan, y la ausencia de índices y el uso de materializaciones incrementan los costos y la complejidad con el aumento del tamaño de los datos.
  - **Con índices por defecto:** Los planes de consulta para las tablas de 1k, 10k, 100k y 1M datos muestran diferencias significativas en los algoritmos utilizados, aunque comparten algunas similitudes clave. En todos los casos, se utiliza un Nested Loop Left Join, Hash Join, Seq Scan en colaborador, SubQuery Scan para filtrar por sede, y un Index Only Scan en profesor. En 1k datos, el plan comienza con un Nested Loop Left Join seguido de un Hash Join entre colaborador y persona, aplicando un Seq Scan en colaborador y un SubQuery Scan para filtrar por sede, con un Index Only Scan en profesor y finalizando con un agregado. Para 10k datos, además de las similitudes mencionadas, se introduce un Bitmap Heap Scan en profesor\_sede, manejando más registros y ajustando las subconsultas de manera más frecuente. En 100k datos, se observa un incremento en los costos debido al mayor volumen de datos, manteniendo las operaciones básicas, pero con un uso intensivo de Bitmap Heap Scan en profesor\_sede y ajustes en las subconsultas. Para 1M datos, se siguen utilizando las mismas operaciones básicas, pero el manejo del volumen de datos masivo se optimiza con un uso más intensivo de Index Only Scan, SubQuery Scan en colaborador y sede, y Bitmap Heap Scan en profesor\_sede, incrementando los costos y manteniendo la eficiencia a través de técnicas consistentes y escalables.
  - **Con índices por defecto más índices personalizados:** Los planes de consulta para las tablas de 1k, 10k, 100k y 1M datos muestran diferencias significativas en los algoritmos utilizados, aunque comparten algunas similitudes clave. En todos los casos, se utiliza un Nested Loop Left Join, Hash Join, Seq Scan en colaborador, SubQuery Scan para sede, y un Index Only Scan en profesor. Para 1k datos, el plan comienza con un Nested Loop Left Join

seguido de un Hash Join entre colaborador y persona, aplicando un Seq Scan en colaborador y un SubQuery Scan para sede, con un Index Only Scan en profesor y finalizando con un agregado. En 10k datos, se introduce un Bitmap Heap Scan en persona usando el nuevo índice, junto con un Bitmap Heap Scan en profesor\_sede. En 100k datos, se mantiene el uso intensivo de Bitmap Heap Scan en persona y profesor\_sede, con costos incrementados debido al mayor volumen de datos. Para 1M datos, el plan incluye un uso intensivo de Bitmap Heap Scan en persona y profesor\_sede, además de Memoize para optimizar el escaneo de sede, incrementando los costos y manteniendo la eficiencia a través de técnicas consistentes y escalables.

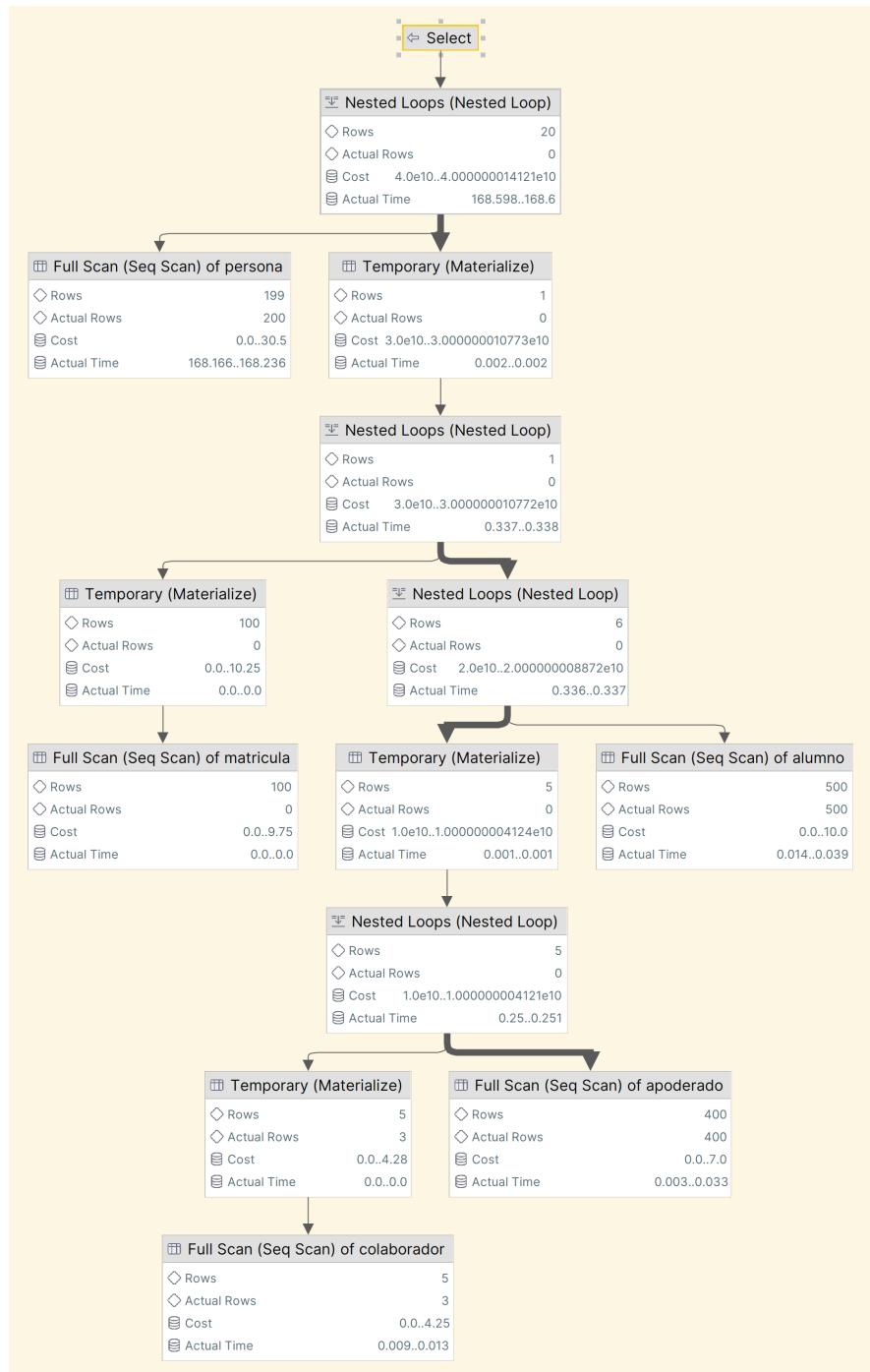
### 5.3.3 Planes de índices para la tercera consulta

- 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_horas_semanales_trabajo;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL apoderado;
13 VACUUM FULL alumno;
14 VACUUM FULL matricula;
15 VACUUM FULL persona;
```

- Para mil datos:



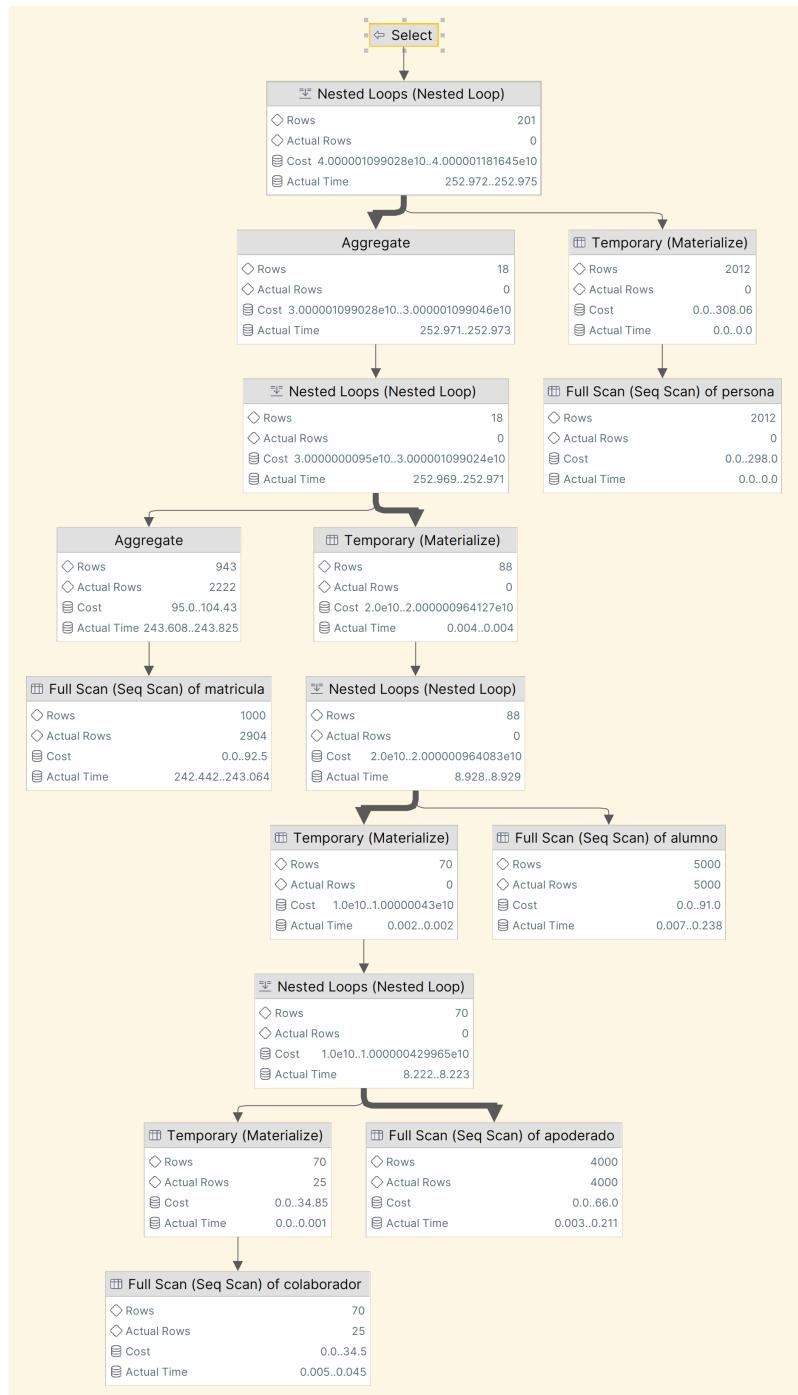
```

Nested Loop Semi Join  (cost=40000000000.00..40000000141.21 rows=20 width=52) (actual time=204.208..204.210 rows=0 loops=1)
  Join Filter: (persona.dni = alumno.dni)
    -> Seq Scan on persona  (cost=0.00..30.50 rows=199 width=61) (actual time=203.841..203.920 rows=200 loops=1)
        Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
        Rows Removed by Filter: 800
    -> Materialize  (cost=30000000000.00..30000000000.00 rows=1 width=18) (actual time=0.001..0.001 rows=0 loops=200)
        -> Nested Loop Semi Join  (cost=30000000000.00..300000000107.72 rows=1 width=18) (actual time=0.263..0.264 rows=0 loops=1)
          Join Filter: (alumno.dni = matricula.alumno_dni)
          -> Nested Loop Semi Join  (cost=20000000000.00..20000000088.72 rows=6 width=9) (actual time=0.262..0.263 rows=0 loops=1)
            Join Filter: (apoderado.dni = alumno.apoderado_dni)
            -> Seq Scan on alumno  (cost=0.00..18 rows=500 width=18) (actual time=0.012..0.037 rows=500 loops=1)
            -> Materialize  (cost=10000000000.00..10000000041.24 rows=5 width=18) (actual time=0.000..0.000 rows=0 loops=500)
                -> Nested Loop  (cost=10000000000.00..10000000041.21 rows=5 width=18) (actual time=0.169..0.169 rows=0 loops=1)
                  Join Filter: (apoderado.dni = colaborador.dni)
                  Rows Removed by Join Filter: 1280
                  -> Seq Scan on apoderado  (cost=0.00..7.00 rows=400 width=9) (actual time=0.007..0.024 rows=400 loops=1)
                  -> Materialize  (cost=0.00..4.28 rows=5 width=9) (actual time=0.000..0.000 rows=3 loops=400)
                      -> Seq Scan on colaborador  (cost=0.00..4.25 rows=5 width=9) (actual time=0.010..0.014 rows=3 loops=1)
                          Filter: (esta_activo AND (horas_semanales_trabajo > 48) AND ((sueldoHora * (horas_semanales_trabajo)::double precision) * '4'::double precision) <= 0)
                          Rows Removed by Filter: 97
                -> Materialize  (cost=0.00..10.25 rows=100 width=9) (never executed)
                    -> Seq Scan on matricula  (cost=0.00..9.75 rows=100 width=9) (never executed)
                        Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))

Planning Time: 0.971 ms
JIT:
  Functions: 30
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.990 ms, Inlining 42.684 ms, Optimization 102.756 ms, Emission 58.358 ms, Total 205.787 ms
Execution Time: 206.292 ms

```

- Para diez mil datos:

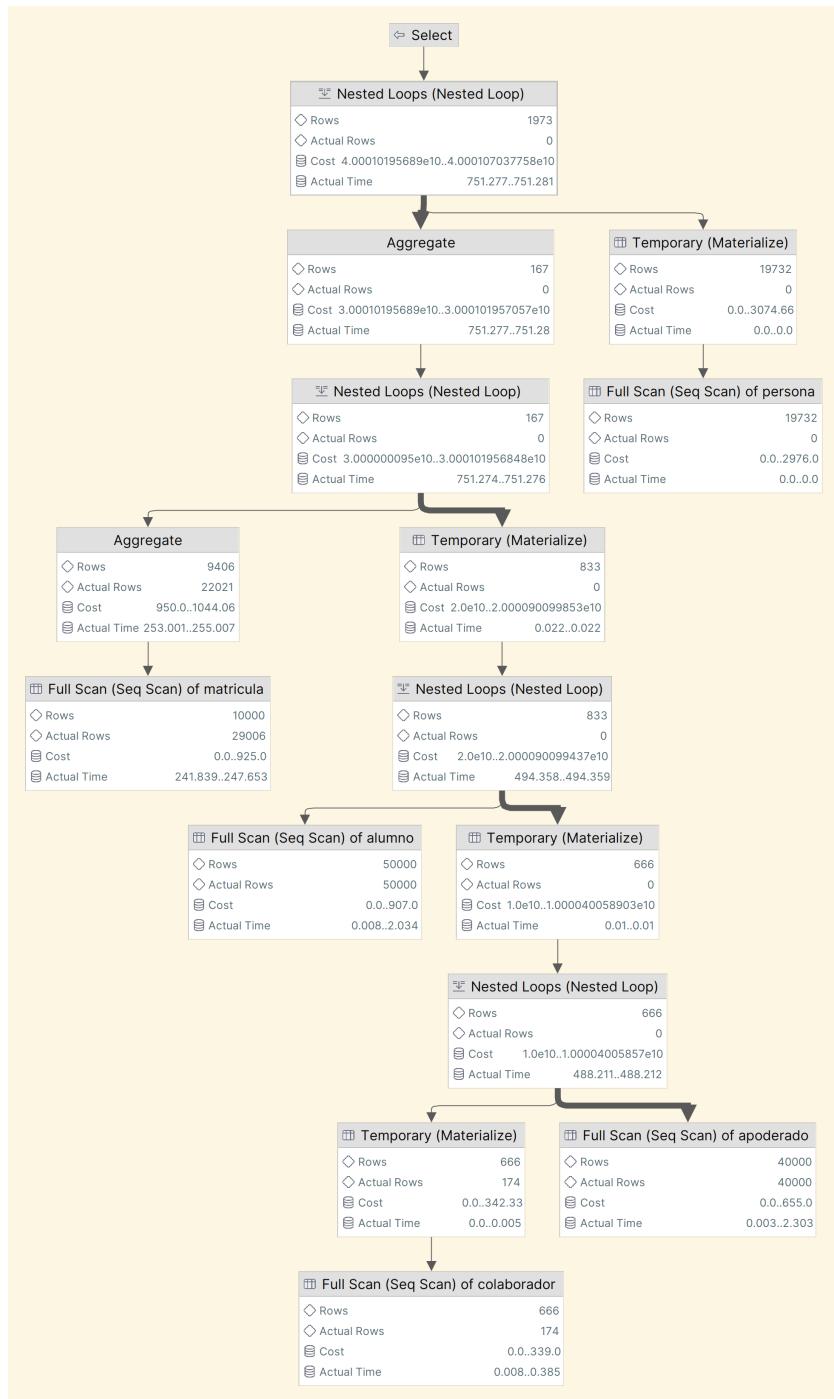


```

Nested Loop  (cost=40000010990.28..40000011816.45 rows=201 width=52) (actual time=302.062..302.065 rows=0 loops=1)
  Join Filter: (persona.dni = alumno.dni)
    -> HashAggregate  (cost=30000010990.28..30000010990.46 rows=18 width=18) (actual time=302.061..302.064 rows=0 loops=1)
      Group Key: alumno.dni
      Batches: 1  Memory Usage: 24kB
        -> Nested Loop  (cost=30000000095.00..300000010990.24 rows=18 width=18) (actual time=302.060..302.062 rows=0 loops=1)
          Join Filter: (alumno.dni = matricula.alumno_dni)
            -> HashAggregate  (cost=95.00..104.43 rows=943 width=9) (actual time=292.968..293.139 rows=2222 loops=1)
              Group Key: matricula.alumno_dni
              Batches: 1  Memory Usage: 241kB
                -> Seq Scan on matricula  (cost=0.00..92.50 rows=1000 width=9) (actual time=291.896..292.471 rows=2984 loops=1)
                  Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                  Rows Removed by Filter: 96
                -> Materialize  (cost=20000000000.00..20000000000.27 rows=88 width=9) (actual time=0.004..0.004 rows=0 loops=2222)
                  -> Nested Loop Semi Join  (cost=20000000000.00..20000000000.83 rows=88 width=9) (actual time=8.724..8.725 rows=0 loops=1)
                    Join Filter: (apoderado.dni = alumno.apoderado_dni)
                    -> Seq Scan on alumno  (cost=0.00..91.00 rows=5000 width=18) (actual time=0.007..0.233 rows=5000 loops=1)
                      -> Materialize  (cost=10000000000.00..10000004300.00 rows=70 width=18) (actual time=0.002..0.002 rows=0 loops=5000)
                        -> Nested Loop  (cost=10000000000.00..10000004299.65 rows=70 width=18) (actual time=8.023..8.024 rows=0 loops=1)
                          Join Filter: (apoderado.dni = colaborador.dni)
                          Rows Removed by Join Filter: 100000
                          -> Seq Scan on apoderado  (cost=0.00..66.00 rows=4000 width=9) (actual time=0.003..0.183 rows=4000 loops=1)
                            -> Materialize  (cost=0.00..34.85 rows=70 width=9) (actual time=0.000..0.001 rows=25 loops=4000)
                              -> Seq Scan on colaborador  (cost=0.00..34.50 rows=70 width=9) (actual time=0.005..0.044 rows=25 loops=1)
                                Filter: (esta_activo AND (horas_semanales_trabajo > 48) AND (((sueldo_hora * (horas_semanales_trabajo)::double precision) * '4'::double precision) > 0))
                                Rows Removed by Filter: 975
                            -> Materialize  (cost=0.00..308.06 rows=2012 width=61) (never executed)
                              -> Seq Scan on persona  (cost=0.00..298.00 rows=2012 width=61) (never executed)
                                Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 0.490 ms
JIT:
  Functions: 49
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.741 ms, Inlining 29.809 ms, Optimization 168.897 ms, Emission 93.167 ms, Total 293.614 ms
Execution Time: 303.995 ms

```

– Para cien mil datos:

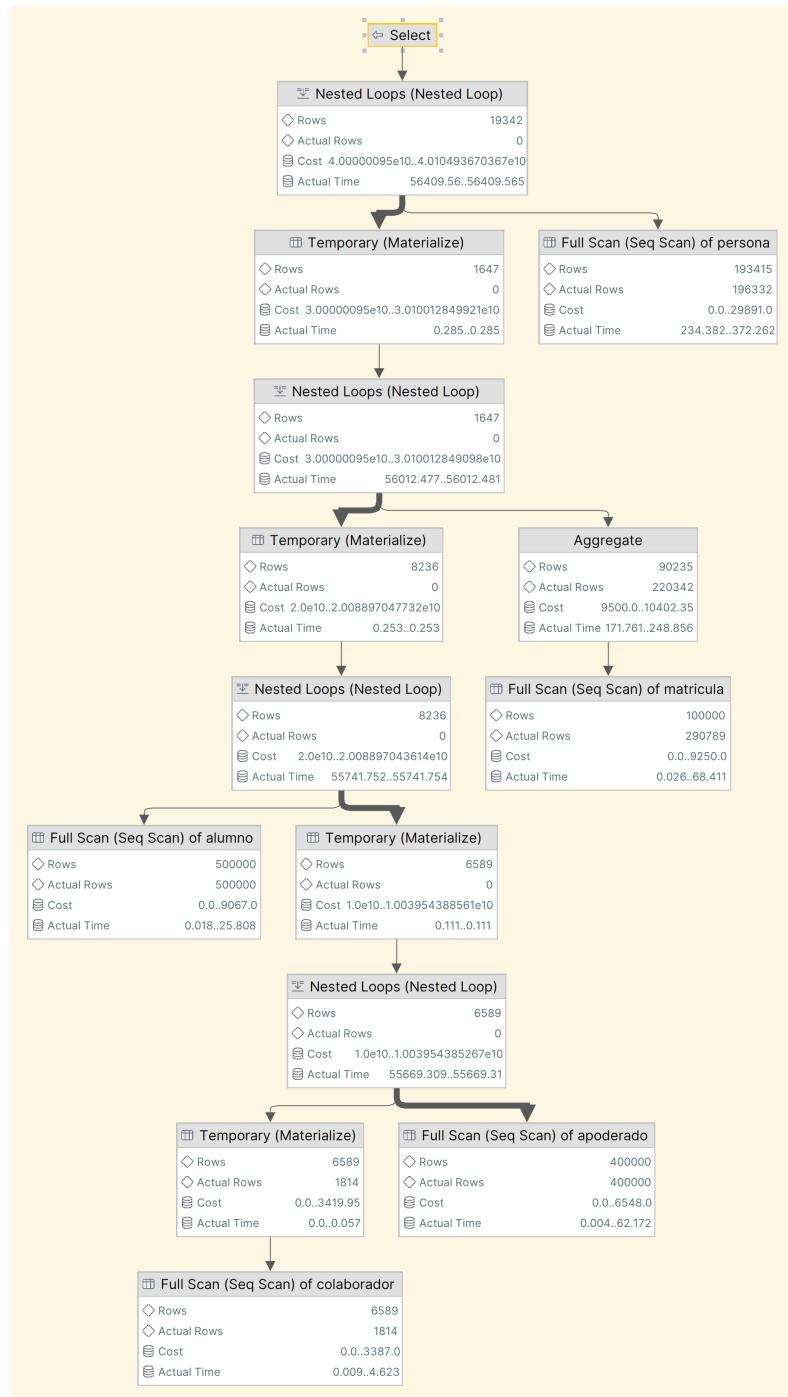


```

Nested Loop  (cost=40001019568.90..40001070386.82 rows=1974 width=53) (actual time=907.009..907.013 rows=0 loops=1)
  Join Filter: (persona.dni = alumno.dni)
    -> HashAggregate  (cost=30001019568.90..30001019570.57 rows=167 width=18) (actual time=907.008..907.012 rows=0 loops=1)
      Group Key: alumno.dni
      Batches: 1  Memory Usage: 40kB
        -> Nested Loop  (cost=300000000950.00..30001019568.48 rows=167 width=18) (actual time=907.005..907.009 rows=0 loops=1)
          Join Filter: (alumno.dni = matricula.alumno.dni)
            -> HashAggregate  (cost=950.00..1044.06 rows=9406 width=9) (actual time=327.976..330.473 rows=22021 loops=1)
              Group Key: matricula.alumno.dni
              Batches: 1  Memory Usage: 1809kB
                -> Seq Scan on matricula  (cost=0.00..925.00 rows=10000 width=9) (actual time=316.265..322.372 rows=29006 loops=1)
                  Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                  Rows Removed by Filter: 994
                -> Materialize  (cost=200000000000.00..200000000098.53 rows=833 width=9) (actual time=0.026..0.026 rows=0 loops=22021)
                  -> Nested Loop Semi Join  (cost=200000000000.00..200000000994.37 rows=833 width=9) (actual time=574.401..574.403 rows=0 loops=1)
                    Join Filter: (apoderado.dni = alumno.apoderado.dni)
                    -> Seq Scan on alumno  (cost=0.00..997.00 rows=50000 width=18) (actual time=0.009..3.269 rows=50000 loops=1)
                    -> Materialize  (cost=10000000000.00..10000400589.03 rows=666 width=18) (actual time=0.011..0.011 rows=0 loops=50000)
                      -> Nested Loop  (cost=10000000000.00..10000400585.70 rows=666 width=18) (actual time=565.104..565.105 rows=0 loops=1)
                        Join Filter: (apoderado.dni = colaborador.dni)
                        Rows Removed by Join Filter: 6960000
                        -> Seq Scan on apoderado  (cost=0.00..655.00 rows=40000 width=9) (actual time=0.003..2.939 rows=40000 loops=1)
                        -> Materialize  (cost=0.00..342.33 rows=666 width=9) (actual time=0.000..0.006 rows=174 loops=40000)
                          -> Seq Scan on colaborador  (cost=0.00..339.00 rows=666 width=9) (actual time=0.007..0.012 rows=174 loops=1)
                            Filter: (esta_activo AND (horas_semanales_trabajo > 48) AND (((sueldo_hora * (horas_semanales_trabajo)::double precision) * '4'::double precision) > 0))
                            Rows Removed by Filter: 9826
                        -> Materialize  (cost=0.00..3074.68 rows=19736 width=62) (never executed)
                          -> Seq Scan on persona  (cost=0.00..2976.00 rows=19736 width=62) (never executed)
                            Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 0.438 ms
JIT:
  Functions: 49
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 2.331 ms, Inlining 27.751 ms, Optimization 185.709 ms, Emission 102.796 ms, Total 318.587 ms
Execution Time: 909.574 ms

```

- Para un millón de datos:



```

Nested Loop Semi Join  (cost=400000009500.00..40104937667.17 rows=19345 width=54) (actual time=57133.059..57133.065 rows=0 loops=1)
  Join Filter: (persona.dni = alumno.dni)
    -> Seq Scan on persona  (cost=0.00..29891.00 rows=193454 width=63) (actual time=264.085..358.022 rows=196382 loops=1)
        Filter: (nacimiento.fecha > (CURRENT_DATE - '18 years'::interval))
        Rows Removed By Filter: 883618
    -> Materialize  (cost=30000009500.00..30100128499.21 rows=1647 width=18) (actual time=0.289..0.289 rows=0 loops=196382)
        -> Nested Loop  (cost=30000009500.00..30100128499.98 rows=1647 width=18) (actual time=56754.226..56754.230 rows=0 loops=1)
          Join Filter: (alumno.dni = matricula.alumno_dni)
            -> HashAggregate  (cost=9500.00..10402.35 rows=90235 width=9) (actual time=145.599..224.686 rows=220342 loops=1)
              Group Key: matricula.alumno_dni
              Batches: 5 Memory Usage: 10289kB Disk Usage: 6968kB
            -> Seq Scan on matricula  (cost=0.00..9250.00 rows=100000 width=9) (actual time=0.030..59.611 rows=290789 loops=1)
                Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                Rows Removed By Filter: 9211
            -> Materialize  (cost=20000000000.00..20888970477.32 rows=8236 width=9) (actual time=0.256..0.256 rows=0 loops=20342)
                -> Nested Loop Semi Join  (cost=20000000000.00..20888970436.14 rows=8236 width=9) (actual time=56504.304..56504.306 rows=0 loops=1)
                  Join Filter: (apoderado.dni = alumno.apoderado_dni)
                    -> Seq Scan on alumno  (cost=0.00..9667.00 rows=500000 width=18) (actual time=0.007..23.249 rows=500000 loops=1)
                    -> Materialize  (cost=10000000000.00..10039543885.61 rows=6589 width=18) (actual time=0.113..0.113 rows=0 loops=500000)
                      -> Nested Loop  (cost=10000000000.00..10039543852.67 rows=6589 width=18) (actual time=56433.718..56433.719 rows=0 loops=1)
                        Join Filter: (apoderado.dni = colaborador.dni)
                        Rows Removed By Filter: 725600000
                        -> Seq Scan on apoderado  (cost=0.00..6548.00 rows=400000 width=9) (actual time=0.004..72.341 rows=400000 loops=1)
                        -> Materialize  (cost=0.00..3419.95 rows=6589 width=9) (actual time=0.000..0.058 rows=1814 loops=400000)
                          -> Seq Scan on colaborador  (cost=0.00..3387.00 rows=6589 width=9) (actual time=0.007..3.882 rows=1814 loops=1)
                            Filter: (esta_activo AND (horas_semanales_trabajo > 48) AND ((sueldo_hora * (horas_semanales_trabajo)::double precision) * '4'::double precision) > 0)
                            Rows Removed By Filter: 98186
Planning Time: 0.477 ms
JIT:
  Functions: 36
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 2.165 ms, Inlining 45.444 ms, Optimization 149.912 ms, Emission 76.131 ms, Total 273.652 ms
Execution Time: 57136.121 ms

```

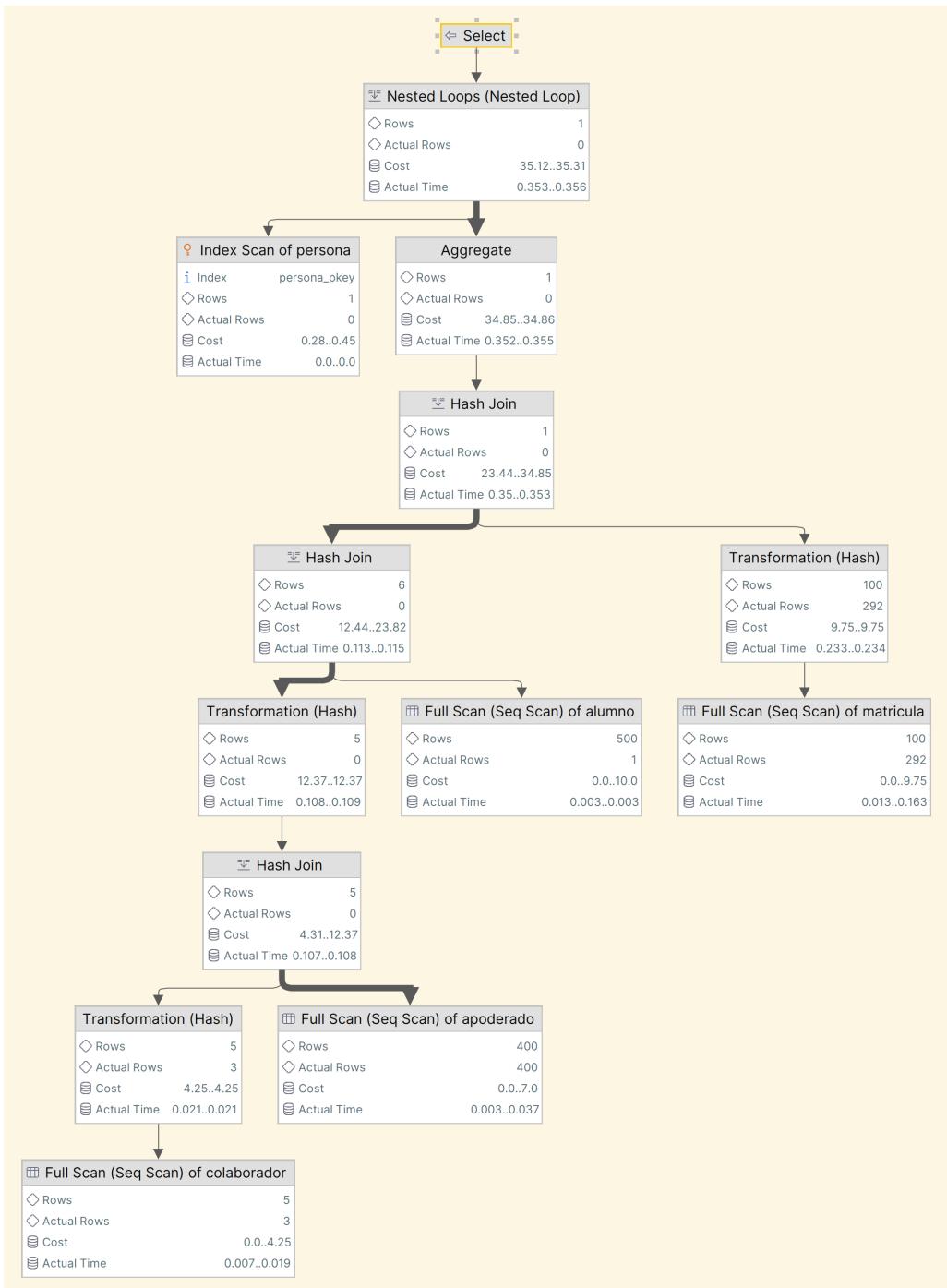
- Ejecución con índices por defecto

```

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_horas_semanales_trabajo;
10
11 VACUUM FULL colaborador;
12 VACUUM FULL apoderado;
13 VACUUM FULL alumno;
14 VACUUM FULL matricula;
15 VACUUM FULL persona;

```

– Para mil datos:

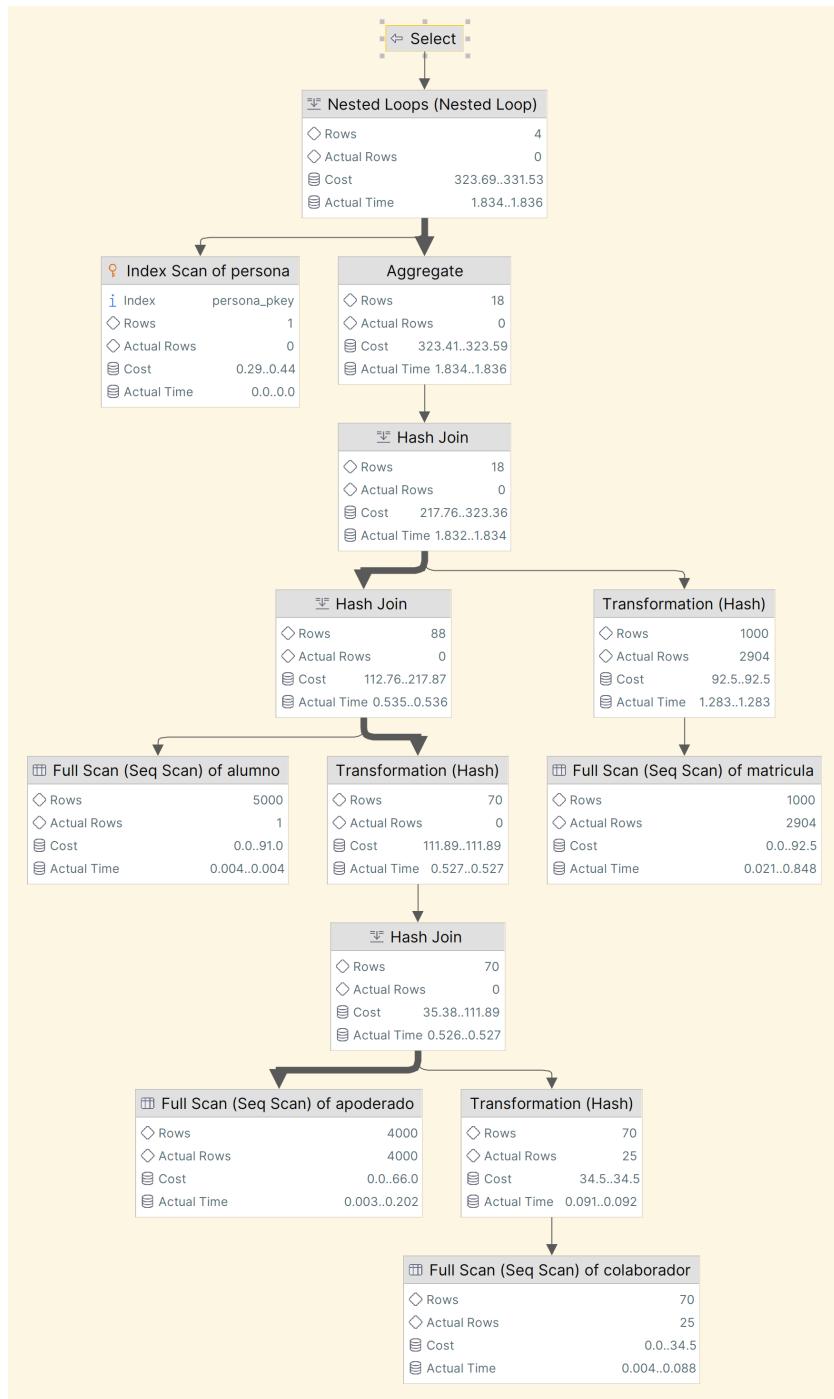


```

Nested Loop  (cost=35.12..35.31  rows=1 width=52) (actual time=0.299..0.302 rows=0 loops=1)
  -> HashAggregate  (cost=34.85..34.86 rows=1 width=18) (actual time=0.299..0.301 rows=0 loops=1)
      Group Key: alumno.dni
      Batches: 1  Memory Usage: 24kB
      -> Hash Semi Join  (cost=23.44..34.85 rows=1 width=18) (actual time=0.297..0.299 rows=0 loops=1)
          Hash Cond: (alumno.dni = matricula.alumno_dni)
          -> Hash Semi Join  (cost=12.44..23.82 rows=6 width=9) (actual time=0.090..0.093 rows=0 loops=1)
              Hash Cond: (alumno.apoderado_dni = apoderado.dni)
              -> Seq Scan on alumno  (cost=0.00..10.00 rows=580 width=18) (actual time=0.002..0.003 rows=1 loops=1)
                  Hash Cond: (alumno.apoderado_dni = apoderado.dni)
                  -> Hash  (cost=12.37..12.37 rows=5 width=18) (actual time=0.087..0.088 rows=0 loops=1)
                      Buckets: 1024  Batches: 1  Memory Usage: 8kB
                      -> Hash Join  (cost=4.31..12.37 rows=5 width=18) (actual time=0.087..0.088 rows=0 loops=1)
                          Hash Cond: (apoderado.dni = colaborador.dni)
                          -> Seq Scan on apoderado  (cost=0.00..7.00 rows=400 width=9) (actual time=0.006..0.038 rows=400 loops=1)
                          -> Hash  (cost=4.25..4.25 rows=5 width=9) (actual time=0.014..0.014 rows=3 loops=1)
                              Buckets: 1024  Batches: 1  Memory Usage: 9kB
                              -> Seq Scan on colaborador  (cost=0.00..4.25 rows=5 width=9) (actual time=0.005..0.013 rows=3 loops=1)
                                  Filter: (esta.activo AND (horas_semanales_trabajo > 48) AND ((sueldoHora * (horas_semanales_trabajo)::double precision) * '4'::double precision) <= 0.0001)
                                  Rows Removed by Filter: 97
                      -> Hash  (cost=9.75..9.75 rows=100 width=9) (actual time=0.203..0.203 rows=292 loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 20kB
                          -> Seq Scan on matricula  (cost=0.00..9.75 rows=100 width=9) (actual time=0.012..0.141 rows=292 loops=1)
                              Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                              Rows Removed by Filter: 8
                      -> Hash  (cost=9.75..9.75 rows=1 width=61) (never executed)
                          Index Cond: (dni = alumno.dni)
                          Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 0.787 ms
Execution Time: 0.351 ms

```

- Para diez mil datos:

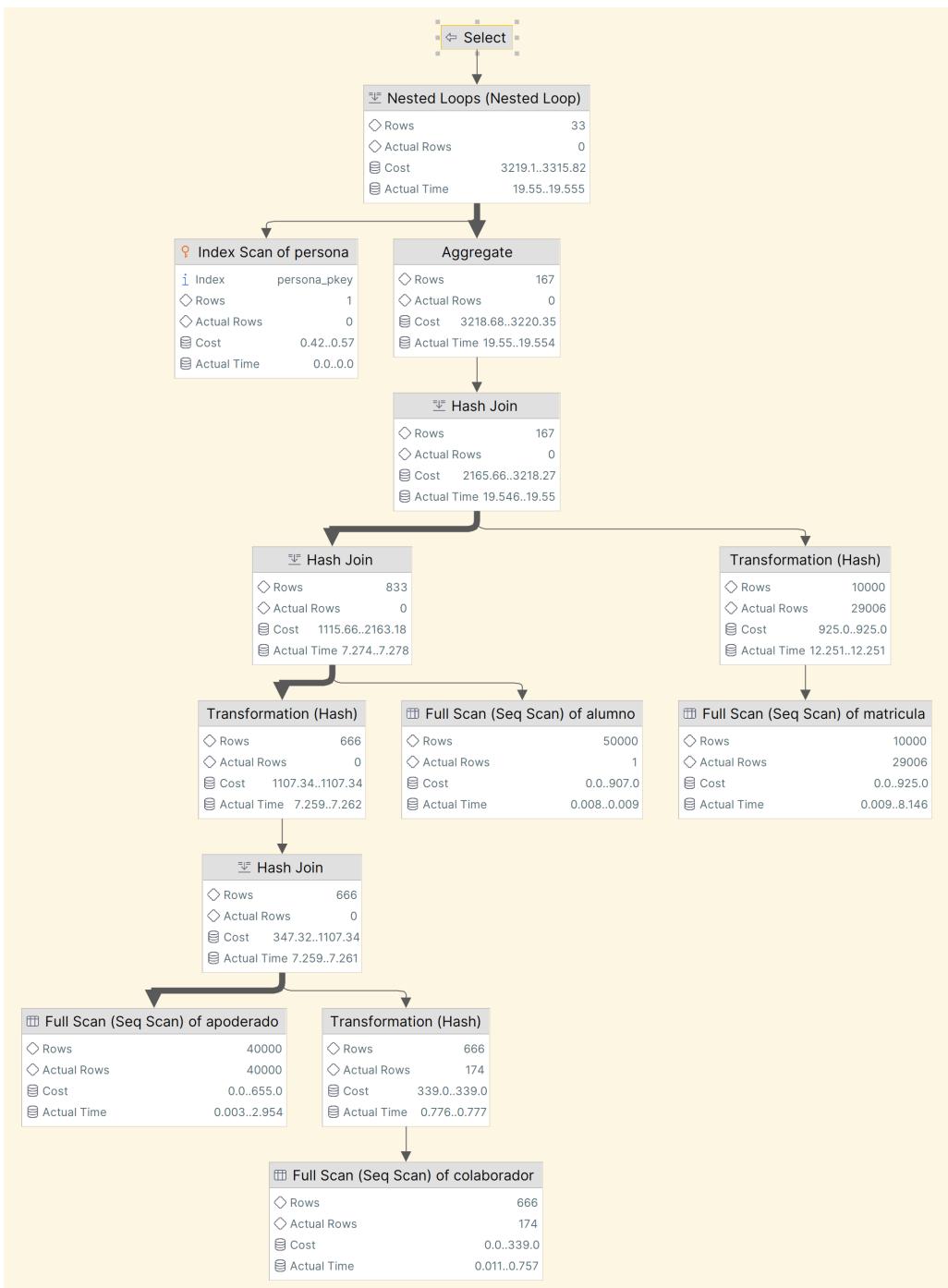


```

Nested Loop  (cost=323.69..331.53 rows=4 width=52) (actual time=2.845..2.848 rows=0 loops=1)
  -> HashAggregate  (cost=323.41..323.59 rows=18 width=18) (actual time=2.844..2.847 rows=0 loops=1)
      Group Key: alumno.dni
      Batches: 1 Memory Usage: 24kB
      -> Hash Semi Join  (cost=217.76..323.36 rows=18 width=18) (actual time=2.842..2.844 rows=0 loops=1)
          Hash Cond: (alumno.dni = matricula.alumno_dni)
          -> Hash Semi Join  (cost=112.76..217.87 rows=88 width=9) (actual time=0.884..0.886 rows=0 loops=1)
              Hash Cond: (alumno.apoderado_dni = apoderado.dni)
              -> Seq Scan on alumno  (cost=0.00..91.00 rows=5800 width=18) (actual time=0.006..0.006 rows=1 loops=1)
              -> Hash  (cost=111.89..111.89 rows=70 width=18) (actual time=0.871..0.873 rows=0 loops=1)
                  Buckets: 1024  Batches: 1 Memory Usage: 8kB
                  -> Hash Join  (cost=35.38..111.89 rows=70 width=18) (actual time=0.871..0.872 rows=0 loops=1)
                      Hash Cond: (apoderado.dni = colaborador.dni)
                      -> Seq Scan on apoderado  (cost=0.00..66.00 rows=4000 width=9) (actual time=0.004..0.327 rows=4000 loops=1)
                      -> Hash  (cost=34.50..34.50 rows=70 width=9) (actual time=0.142..0.143 rows=25 loops=1)
                          Buckets: 1024  Batches: 1 Memory Usage: 10kB
                          -> Seq Scan on colaborador  (cost=0.00..34.50 rows=70 width=9) (actual time=0.008..0.136 rows=25 loops=1)
                              Filter: (esta.activo AND (horas_semanales_trabajo > 48) AND ((sueldoHora * (horas_semanales_trabajo)::double precision) * '4'::double precision) <= 0.0001)
                              Rows Removed by Filter: 975
                  -> Hash  (cost=92.50..92.50 rows=1000 width=9) (actual time=1.951..1.952 rows=2984 loops=1)
                      Buckets: 4096 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 149kB
                      -> Seq Scan on matricula  (cost=0.00..92.50 rows=1000 width=9) (actual time=0.027..1.366 rows=2984 loops=1)
                          Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                          Rows Removed by Filter: 96
                  -> Index Scan using persona_pkey on persona  (cost=0.29..0.44 rows=1 width=61) (never executed)
                      Index Cond: (dni = alumno.dni)
                      Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 1.543 ms
Execution Time: 2.979 ms

```

- Para cien mil datos:

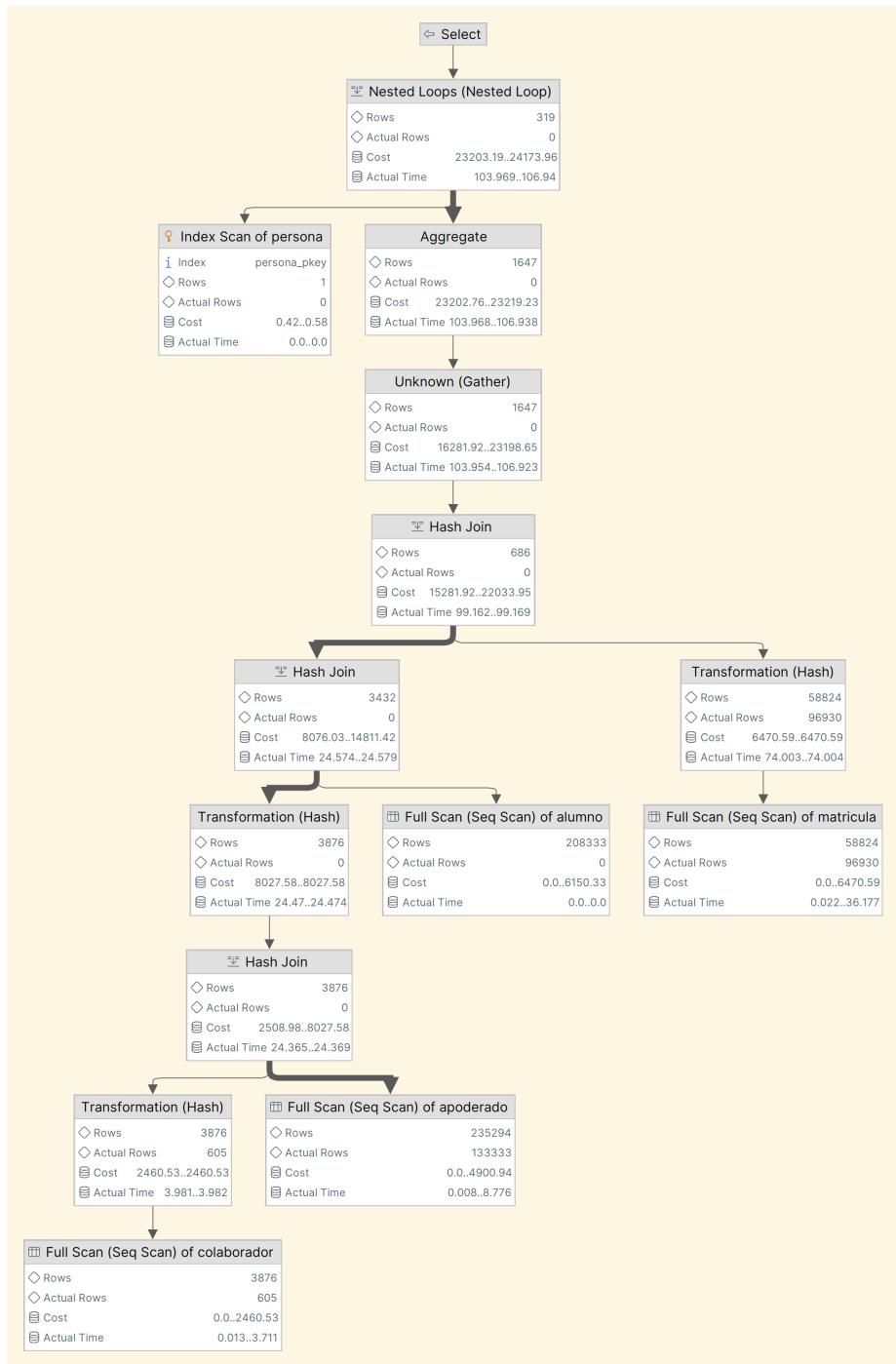


```

Nested Loop  (cost=3219.10..3315.82 rows=33 width=53) (actual time=23.976..23.980 rows=0 loops=1)
  -> HashAggregate  (cost=3218.68..3220.35 rows=167 width=18) (actual time=23.975..23.979 rows=0 loops=1)
    Group Key: alumno.dni
    Batches: 1 Memory Usage: 40kB
    -> Hash Semi Join  (cost=2165.66..3218.27 rows=167 width=18) (actual time=23.971..23.975 rows=0 loops=1)
      Hash Cond: (alumno.dni = matricula.alumno_dni)
      -> Hash Semi Join  (cost=1115.66..2163.18 rows=833 width=9) (actual time=5.674..5.677 rows=0 loops=1)
        Hash Cond: (alumno.apoderado_dni = apoderado.dni)
        -> Seq Scan on alumno  (cost=0.00..997.00 rows=50000 width=18) (actual time=0.014..0.014 rows=1 loops=1)
        -> Hash  (cost=1107.34..1107.34 rows=666 width=18) (actual time=5.651..5.653 rows=0 loops=1)
          Buckets: 1024 Batches: 1 Memory Usage: 8kB
          -> Hash Join  (cost=347.32..1107.34 rows=666 width=18) (actual time=5.650..5.652 rows=0 loops=1)
            Hash Cond: (apoderado.dni = colaborador.dni)
            -> Seq Scan on apoderado  (cost=0.00..655.00 rows=40000 width=9) (actual time=0.003..1.980 rows=40000 loops=1)
            -> Hash  (cost=339.00..339.00 rows=666 width=9) (actual time=1.066..1.067 rows=174 loops=1)
              Buckets: 1024 Batches: 1 Memory Usage: 15kB
              -> Seq Scan on colaborador  (cost=0.00..339.00 rows=666 width=9) (actual time=0.007..1.040 rows=174 loops=1)
                Filter: (esta.activo AND (horas_semanales_trabajo > 48) AND ((sueldoHora * (horas_semanales_trabajo)::double precision) * '4'::double precision) <= 925.00)
                Rows Removed by Filter: 9826
            -> Hash  (cost=925.00..925.00 rows=100000 width=9) (actual time=18.284..18.284 rows=29006 loops=1)
              Buckets: 32768 (originally 16384) Batches: 1 (originally 1) Memory Usage: 1418kB
              -> Seq Scan on matricula  (cost=0.00..925.00 rows=10000 width=9) (actual time=0.008..13.336 rows=29006 loops=1)
                Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                Rows Removed by Filter: 994
            -> Index Scan using persona_pkkey on persona  (cost=0.42..0.57 rows=1 width=62) (never executed)
              Index Cond: (dni = alumno.dni)
              Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 1.126 ms
Execution Time: 24.037 ms

```

- Para un millón de datos:



```

Nested Loop  (cost=23203.19..24173.96 rows=319 width=54) (actual time=123.492..125.516 rows=0 loops=1)
  -> HashAggregate  (cost=23202.76..23219.23 rows=1647 width=18) (actual time=123.491..125.514 rows=0 loops=1)
    Group Key: alumno.dni
    Batches: 1 Memory Usage: 73kB
    -> Gather  (cost=16281.92..23198.65 rows=1647 width=18) (actual time=123.449..125.471 rows=0 loops=1)
      Workers Planned: 2
      Workers Launched: 2
      -> Parallel Hash Semi Join  (cost=15281.92..22033.95 rows=686 width=18) (actual time=116.116..116.121 rows=0 loops=3)
        Hash Cond: (alumno.dni = matricula.alumno.dni)
        -> Parallel Hash Semi Join  (cost=8076.03..14811.42 rows=3432 width=9) (actual time=24.312..24.315 rows=0 loops=3)
          Hash Cond: (alumno.apoderado.dni = apoderado.dni)
          -> Parallel Seq Scan on alumno  (cost=0.00..6150.33 rows=208333 width=18) (never executed)
          -> Parallel Hash  (cost=8027.58..8027.58 rows=3876 width=18) (actual time=24.135..24.137 rows=0 loops=3)
            Buckets: 8192 Batches: 1 Memory Usage: 0kB
            -> Parallel Hash Join  (cost=2968.98..8827.58 rows=3876 width=18) (actual time=24.012..24.014 rows=0 loops=3)
              Hash Cond: (apoderado.dni = colaborador.dni)
              -> Parallel Seq Scan on apoderado  (cost=0.00..4900.94 rows=235294 width=9) (actual time=0.009..8.585 rows=133333 loops=3)
              -> Parallel Hash  (cost=2460.53..2460.53 rows=3876 width=9) (actual time=4.213..4.214 rows=605 loops=3)
                Buckets: 8192 Batches: 1 Memory Usage: 192kB
                -> Parallel Seq Scan on colaborador  (cost=0.00..2460.53 rows=3876 width=9) (actual time=0.020..4.018 rows=605 loops=3)
                  Filter: (esta_activo AND (horas_semanales_trabajo > 48) AND (((sueldo_hora * horas_semanales_trabajo)::double precision) * '4'::double
                    Rows Removed by Filter: 32729
              -> Parallel Hash  (cost=6470.59..6470.59 rows=58824 width=9) (actual time=90.599..90.600 rows=96930 loops=3)
                Buckets: 524288 (originally 131072) Batches: 1 (originally 1) Memory Usage: 20864kB
                -> Parallel Seq Scan on matricula  (cost=0.00..6470.59 rows=58824 width=9) (actual time=0.022..44.562 rows=96930 loops=3)
                  Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '10 years'::interval))
                  Rows Removed by Filter: 3070
    -> Index Scan using persona_pkey on persona  (cost=0.42..0.58 rows=1 width=63) (never executed)
      Index Cond: (dni = alumno.dni)
      Filter: (nacimiento_fecha > (CURRENT_DATE - '10 years'::interval))
Planning Time: 1.274 ms
Execution Time: 125.580 ms

```

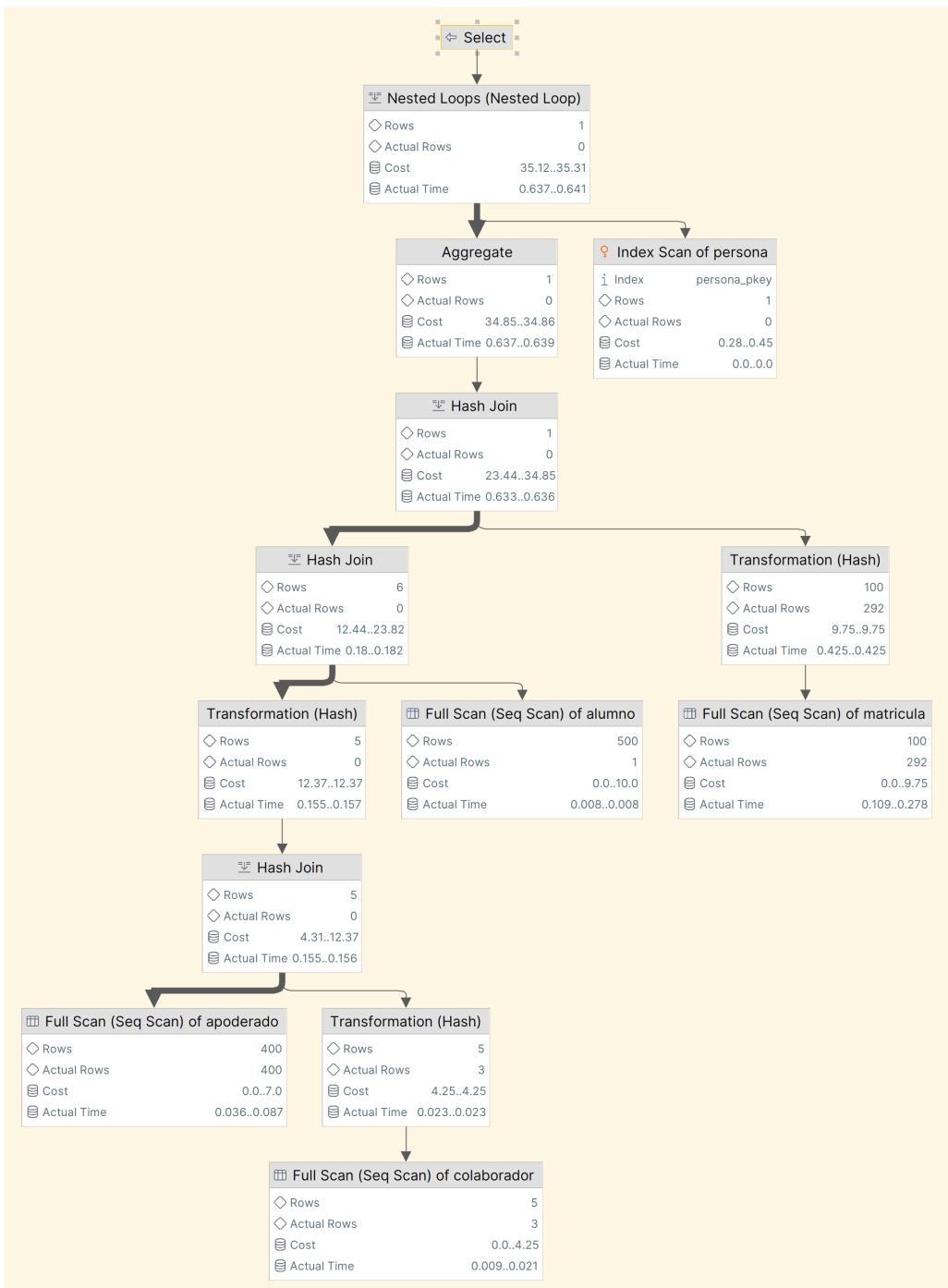
- Ejecución con índices por defecto más índices personalizados

```

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 VACUUM FULL colaborador;
10 VACUUM FULL apoderado;
11 VACUUM FULL alumno;
12 VACUUM FULL matricula;
13 VACUUM FULL persona;
14
15 CREATE INDEX IF NOT EXISTS idx_horas_semanales_trabajo ON colaborador (
  horas_semanales_trabajo);

```

- Para mil datos:

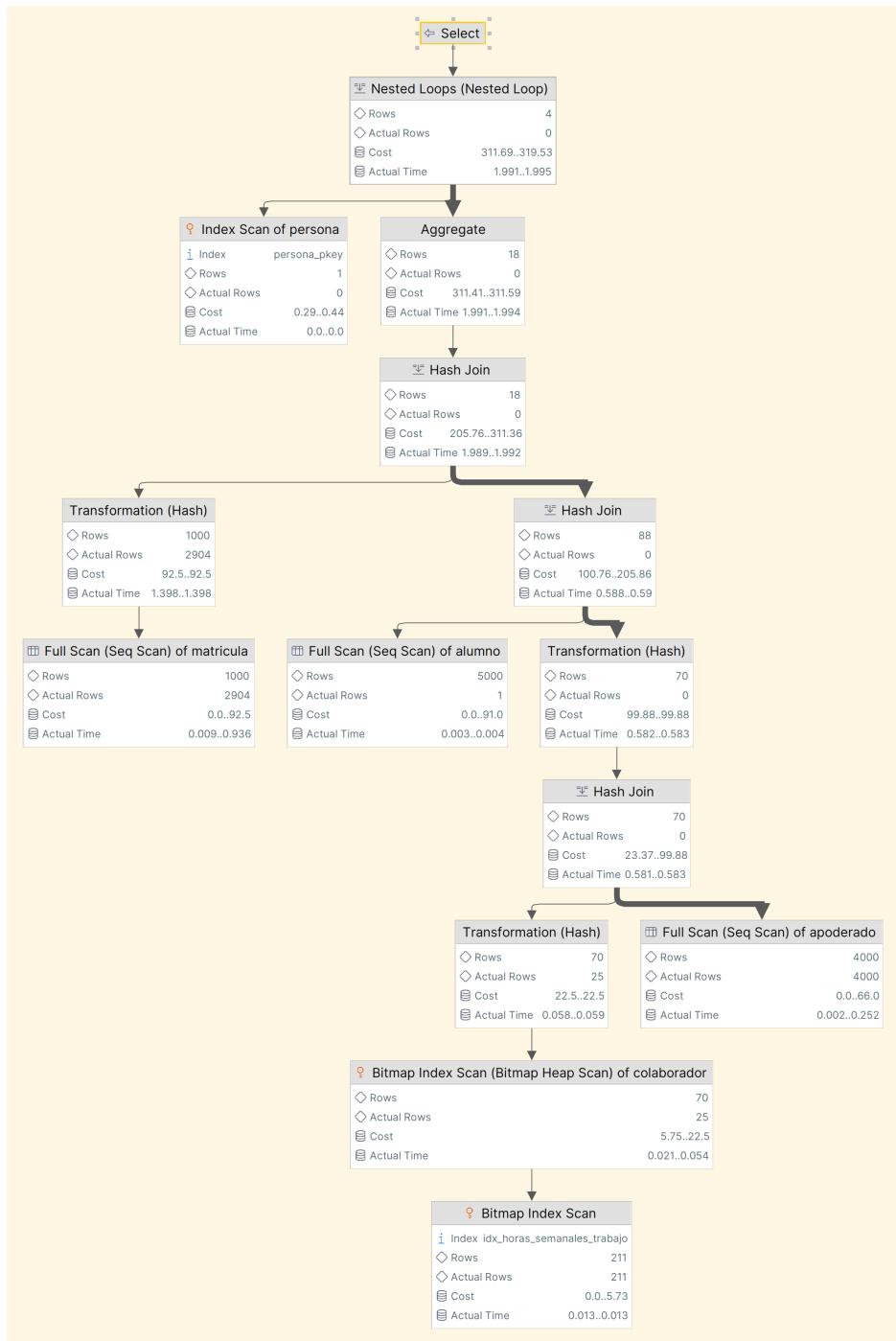


```

Nested Loop  (cost=35.12..35.31  rows=1 width=52) (actual time=0.222..0.224  rows=0 loops=1)
  -> HashAggregate  (cost=34.85..34.86  rows=1 width=18) (actual time=0.221..0.224  rows=0 loops=1)
      Group Key: alumno.dni
      Batches: 1  Memory Usage: 24kB
      -> Hash Semi Join  (cost=23.44..34.85  rows=1 width=18) (actual time=0.199..0.201  rows=0 loops=1)
          Hash Cond: (alumno.dni = matricula.alumno_dni)
          -> Hash Semi Join  (cost=12.44..23.82  rows=6 width=9) (actual time=0.064..0.066  rows=0 loops=1)
              Hash Cond: (alumno.apoderado_dni = apoderado.dni)
              -> Seq Scan on alumno  (cost=0.00..10.00  rows=580  width=18) (actual time=0.002..0.002  rows=1  loops=1)
              -> Hash  (cost=12.37..12.37  rows=5  width=18) (actual time=0.061..0.062  rows=0 loops=1)
                  Buckets: 1024  Batches: 1  Memory Usage: 8kB
                  -> Hash Join  (cost=4.31..12.37  rows=5  width=18) (actual time=0.061..0.062  rows=0 loops=1)
                      Hash Cond: (apoderado.dni = colaborador.dni)
                      -> Seq Scan on apoderado  (cost=0.00..7.00  rows=400  width=9) (actual time=0.002..0.020  rows=400  loops=1)
                      -> Hash  (cost=4.25..4.25  rows=5  width=9) (actual time=0.015..0.015  rows=3  loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 9kB
                          -> Seq Scan on colaborador  (cost=0.00..4.25  rows=5  width=9) (actual time=0.004..0.014  rows=3  loops=1)
                              Filter: (esta.activo AND (horas_semanales_trabajo > 48) AND (((sueldoHora * (horas_semanales_trabajo)::double precision) * '4'::double precision) <= 0.0001))
                              Rows Removed by Filter: 97
                  -> Hash  (cost=9.75..9.75  rows=100  width=9) (actual time=0.132..0.132  rows=292  loops=1)
                      Buckets: 1024  Batches: 1  Memory Usage: 20kB
                      -> Seq Scan on matricula  (cost=0.00..9.75  rows=100  width=9) (actual time=0.008..0.002  rows=292  loops=1)
                          Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                          Rows Removed by Filter: 8
                  -> Index Scan using persona_pkkey on persona  (cost=0.28..0.45  rows=1 width=61) (never executed)
                      Index Cond: (dni = alumno.dni)
                      Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 0.716 ms
Execution Time: 0.262 ms

```

- Para diez mil datos:

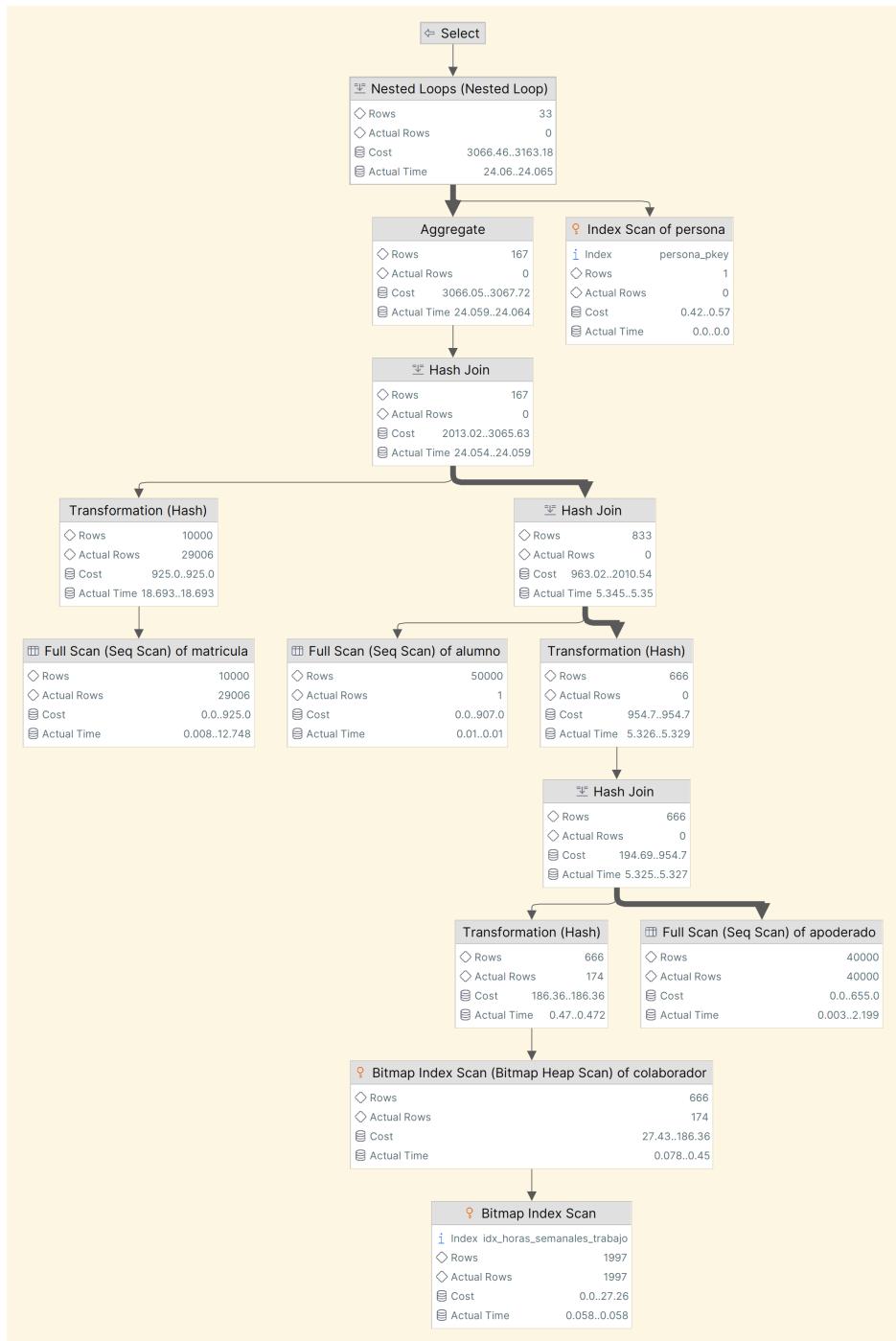


```

Nested Loop  (cost=311.69..319.53 rows=4 width=52) (actual time=1.764..1.767 rows=0 loops=1)
  -> HashAggregate  (cost=311.41..311.59 rows=18 width=18) (actual time=1.764..1.766 rows=0 loops=1)
      Group Key: alumno.dni
      Batches: 1  Memory Usage: 24kB
      -> Hash Semi Join  (cost=205.76..311.36 rows=18 width=18) (actual time=1.762..1.764 rows=0 loops=1)
          Hash Cond: (alumno.dni = matricula.alumno_dni)
          -> Hash Semi Join  (cost=100.76..205.86 rows=88 width=9) (actual time=0.517..0.518 rows=0 loops=1)
              Hash Cond: (alumno.apoderado_dni = apoderado.dni)
              -> Seq Scan on alumno  (cost=0.00..91.00 rows=5000 width=18) (actual time=0.003..0.004 rows=1 loops=1)
                  -> Hash  (cost=99.88..99.88 rows=70 width=18) (actual time=0.510..0.511 rows=0 loops=1)
                      Buckets: 1024  Batches: 1  Memory Usage: 8kB
                      -> Hash Join  (cost=23.37..99.88 rows=70 width=18) (actual time=0.510..0.511 rows=0 loops=1)
                          Hash Cond: (apoderado.dni = colaborador.dni)
                          -> Seq Scan on apoderado  (cost=0.00..66.00 rows=4000 width=9) (actual time=0.002..0.206 rows=4000 loops=1)
                          -> Hash  (cost=22.50..22.50 rows=70 width=9) (actual time=0.062..0.062 rows=25 loops=1)
                              Buckets: 1024  Batches: 1  Memory Usage: 10kB
                              -> Bitmap Heap Scan on colaborador  (cost=5.75..22.50 rows=70 width=9) (actual time=0.020..0.058 rows=25 loops=1)
                                  Recheck Cond: (horas_semanales_trabajo > 48)
                                  Filter: (esta_activa AND (((sueldoHora * (horas_semanales_trabajo)::double precision) * '4'::double precision) < '2000'::double precision))
                                  Rows Removed by Filter: 186
                                  Heap Blocks: exact=12
                                  -> Bitmap Index Scan on idx_horas_semanales_trabajo  (cost=0.00..5.73 rows=211 width=0) (actual time=0.013..0.013 rows=211 loops=1)
                                      Index Cond: (horas_semanales_trabajo > 48)
              -> Hash  (cost=92.50..92.50 rows=1000 width=9) (actual time=1.242..1.243 rows=2904 loops=1)
                  Buckets: 4096 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 149kB
                  -> Seq Scan on matricula  (cost=0.00..92.50 rows=1000 width=9) (actual time=0.008..0.008 rows=2994 loops=1)
                      Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                      Rows Removed by Filter: 96
          -> Index Scan using persona_pkey on persona  (cost=0.29..0.44 rows=1 width=61) (never executed)
              Index Cond: (dni = alumno.dni)
              Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 0.671 ms
Execution Time: 1.812 ms

```

- Para cien mil datos:

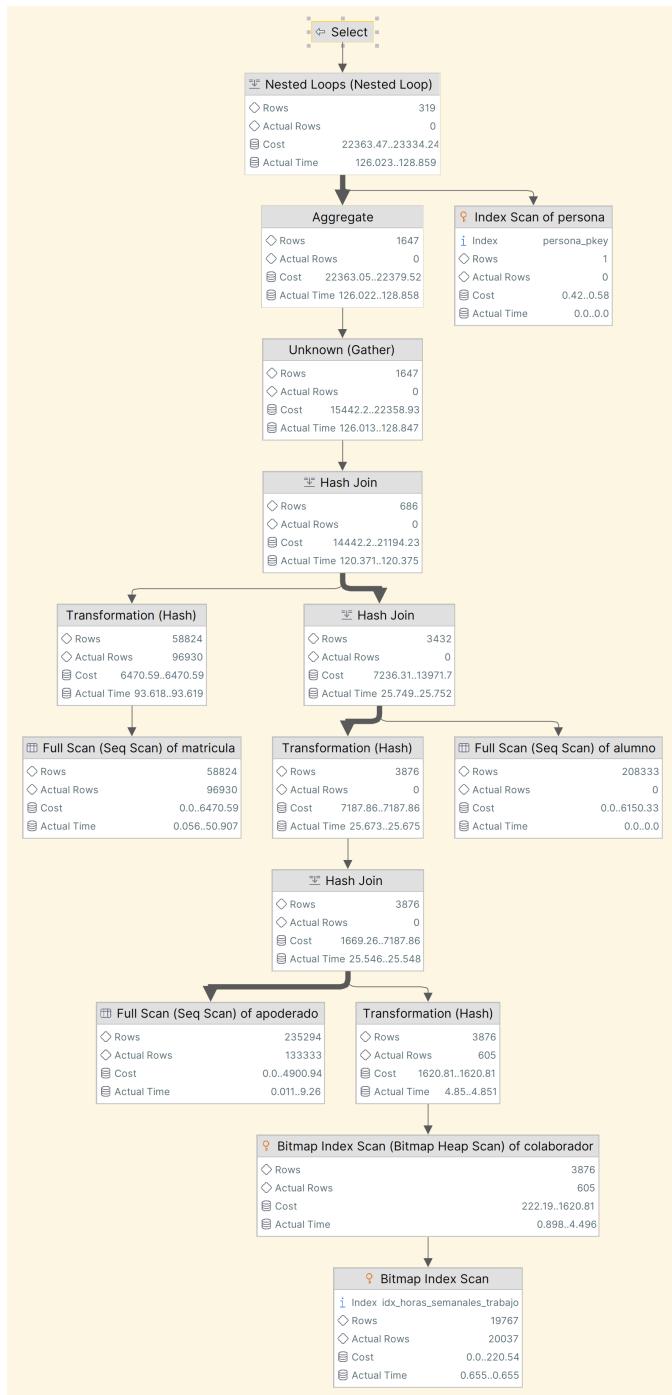


```

Nested Loop  (cost=3066.46..3163.18 rows=33 width=53) (actual time=18.150..18.155 rows=0 loops=1)
  -> HashAggregate  (cost=3066.05..3067.72 rows=167 width=18) (actual time=18.149..18.154 rows=0 loops=1)
      Group Key: alumno.dni
      Batches: 1 Memory Usage: 40kB
      -> Hash Semi Join  (cost=2013.02..3065.63 rows=167 width=18) (actual time=18.144..18.149 rows=0 loops=1)
          Hash Cond: (alumno.dni = matricula.alumno_dni)
          -> Hash Semi Join  (cost=963.02..2010.54 rows=833 width=9) (actual time=4.621..4.625 rows=0 loops=1)
              Hash Cond: (alumno.apoderado_dni = apoderado.dni)
              -> Seq Scan on alumno  (cost=0.00..907.00 rows=50000 width=18) (actual time=0.013..0.014 rows=1 loops=1)
              -> Hash  (cost=954.70..954.70 rows=666 width=18) (actual time=4.601..4.603 rows=0 loops=1)
                  Hash Cond: (alumno.apoderado_dni = apoderado.dni)
                  -> Hash Join  (cost=194.69..954.70 rows=666 width=18) (actual time=4.600..4.602 rows=0 loops=1)
                      Hash Cond: (apoderado.dni = colaborador.dni)
                      -> Seq Scan on apoderado  (cost=0.00..655.00 rows=40000 width=9) (actual time=0.003..1.866 rows=40000 loops=1)
                      -> Hash  (cost=186.36..186.36 rows=666 width=9) (actual time=0.392..0.393 rows=174 loops=1)
                          Buckets: 1024 Batches: 1 Memory Usage: 15kB
                          -> Bitmap Heap Scan on colaborador  (cost=27.43..186.36 rows=666 width=9) (actual time=0.077..0.374 rows=174 loops=1)
                              Recheck Cond: (horas_semanales_trabajo > 48)
                              Filter: (esta_activa AND (((sueldo_hora * (horas_semanales_trabajo)::double precision) * '4'::double precision) < '2000'::double precision))
                              Rows Removed by Filter: 1823
                              Heap Blocks: exact=114
                          -> Bitmap Index Scan on idx_horas_semanales_trabajo  (cost=0.00..27.26 rows=1997 width=0) (actual time=0.059..0.060 rows=1997 loops=1)
                              Index Cond: (horas_semanales_trabajo > 48)
          -> Hash  (cost=925.00..925.00 rows=10000 width=9) (actual time=13.512..13.512 rows=29006 loops=1)
              Buckets: 32768 (originally 16384) Batches: 1 (originally 1) Memory Usage: 1418kB
              -> Seq Scan on matricula  (cost=0.00..925.00 rows=10000 width=9) (actual time=0.008..9.198 rows=29006 loops=1)
                  Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                  Rows Removed by Filter: 994
          -> Index Scan using persona_pk on persona  (cost=0.42..0.57 rows=1 width=62) (never executed)
              Index Cond: (dni = alumno.dni)
              Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 0.701 ms
Execution Time: 18.207 ms

```

- Para un millón de datos:



```

Nested Loop  (cost=22363.47 .. 23334.24 rows=319 width=54) (actual time=102.389..105.061 rows=0 loops=1)
  -> HashAggregate  (cost=22363.05 .. 22379.52 rows=1647 width=18) (actual time=102.388..105.059 rows=0 loops=1)
      Group Key: alumno.dni
      Batches: 1 Memory Usage: 73kB
      -> Gather  (cost=15442.20 .. 22358.93 rows=1647 width=18) (actual time=102.378..105.048 rows=0 loops=1)
          Workers Planned: 2
          Workers Launched: 2
          -> Parallel Hash Semi Join  (cost=14442.20 .. 21194.23 rows=686 width=18) (actual time=98.742..98.747 rows=0 loops=3)
              Hash Cond: (alumno.dni = matricula.alumno_dni)
              -> Parallel Hash Semi Join  (cost=7236.31 .. 13971.70 rows=3432 width=9) (actual time=23.064..23.067 rows=0 loops=3)
                  Hash Cond: (alumno.apoderado_dni = apoderado.dni)
                  -> Parallel Seq Scan on alumno  (cost=0.00 .. 6150.33 rows=20833 width=18) (never executed)
                  -> Parallel Hash  (cost=7187.86 .. 7187.86 rows=3876 width=18) (actual time=22.846..22.848 rows=0 loops=3)
                      Buckets: 8192 Batches: 1 Memory Usage: 8kB
                      -> Parallel Hash Join  (cost=1669.26 .. 7187.86 rows=3876 width=18) (actual time=22.707..22.709 rows=0 loops=3)
                          Hash Cond: (apoderado.dni = colaborador.dni)
                          -> Parallel Seq Scan on apoderado  (cost=0.00 .. 4900.94 rows=235294 width=9) (actual time=0.010..9.005 rows=133333 loops=3)
                          -> Parallel Hash  (cost=1620.81 .. 1620.81 rows=3876 width=9) (actual time=2.666..2.667 rows=605 loops=3)
                          Buckets: 8192 Batches: 1 Memory Usage: 192kB
                          -> Parallel Bitmap Heap Scan on colaborador  (cost=222.19 .. 1620.81 rows=3876 width=9) (actual time=0.806..2.498 rows=605 loops=3)
                              Recheck Cond: (horas_semanales_trabajo > 48)
                              Filter: (esta_activo AND ((sueldo_hora * (horas_semanales_trabajo)::double precision) * '4'::double precision) < '2000'::double precision)
                              Rows Removed by Filter: 6074
                              Heap Blocks: exact=457
                          -> Bitmap Index Scan on idx_horas_semanales_trabajo  (cost=0.00 .. 220.54 rows=19767 width=0) (actual time=0.590..0.591 rows=20037 loops=3)
                              Index Cond: (horas_semanales_trabajo > 48)
                          -> Parallel Hash  (cost=6470.59 .. 6470.59 rows=58824 width=9) (actual time=75.020..75.021 rows=96930 loops=3)
                              Buckets: 524288 (originally 131072) Batches: 1 (originally 1) Memory Usage: 20864kB
                              -> Parallel Seq Scan on matricula  (cost=0.00 .. 6470.59 rows=58824 width=9) (actual time=0.016..35.684 rows=96930 loops=3)
                                  Filter: ((year)::numeric <= (EXTRACT(year FROM CURRENT_DATE) - '2'::numeric))
                                  Rows Removed by Filter: 3070
                          -> Index Scan using persona_pkkey on persona  (cost=0.42 .. 0.58 rows=1 width=63) (never executed)
                              Index Cond: (dni = alumno.dni)
                              Filter: (nacimiento_fecha > (CURRENT_DATE - '18 years'::interval))
Planning Time: 0.749 ms
Execution Time: 105.109 ms

```

- Descripción de las ejecuciones

- **Sin índices:** Los planes de consulta para las tablas de 1k, 10k, 100k y 1M datos muestran diferencias significativas en los algoritmos utilizados, aunque comparten algunas similitudes clave. En todos los casos, se utiliza un Nested Loop Semi Join, Seq Scan en persona, y Materialize en varios niveles de subconsultas. Para 1k datos, el plan inicia con un Nested Loop Semi Join entre persona y alumno, seguido de un Seq Scan en persona, un Nested Loop Semi Join entre alumno y matricula, y un Seq Scan en alumno, aplicando filtros y materializando subconsultas para apoderado y colaborador. En 10k datos, se introduce un HashAggregate para agrupar alumno.dni y un Nested Loop Semi Join entre alumno y apoderado, con un Seq Scan en matricula aplicando el filtro de year. En 100k datos, se mantiene el uso de HashAggregate y Nested Loop Semi Join, con un aumento en los costos y filas removidas por los filtros. Para 1M datos, el plan incluye un Nested Loop Semi Join entre persona y alumno, un HashAggregate en matricula para agrupar por alumno\_dni, y un Nested Loop Semi Join entre alumno y apoderado, con materialización de subconsultas y un Seq Scan en colaborador para aplicar los filtros de esta\_activo, horas\_semanales\_trabajo, y sueldo\_hora, demostrando un incremento significativo en costos y uso de recursos.
- **Con índices por defecto:** Los planes de consulta para las tablas de 1k, 10k, 100k y 1M datos muestran diferencias significativas en los algoritmos utilizados, aunque comparten algunas similitudes clave. En todos los casos, se utilizan Nested Loop, HashAggregate, y Hash Semi Join. Para 1k datos, el plan comienza con un Nested Loop entre persona y alumno, seguido de HashAggregate y Hash Semi Join entre alumno y matricula, con un Seq Scan en alumno, un Hash Join entre apoderado y colaborador, y un Index Scan en persona. En 10k datos, se mantiene el mismo enfoque, pero con un aumento en el número de filas y el uso de memoria. En 100k datos, los costos y el uso de recursos aumentan significativamente, pero el flujo de operaciones sigue siendo similar. Para 1M datos, se introduce la paralelización con Gather y Parallel Hash Semi Join, con Parallel Seq Scan en alumno y matricula, y Parallel Hash Join entre apoderado y colaborador, además de un Index Scan en persona, demostrando un incremento significativo en costos y eficiencia en el manejo de grandes volúmenes de datos.
- **Con índices por defecto más índices personalizados:** Los planes de consulta para las tablas de 1k, 10k, 100k y 1M datos muestran diferencias significativas en los algoritmos utilizados, aunque comparten algunas similitudes clave como el uso de Nested Loop, HashAggregate, y Hash Semi Join en todos los planes. Para 1k datos, el plan comienza con un Nested Loop entre persona y alumno, seguido de HashAggregate y Hash Semi Join entre alumno y matricula,

con un Seq Scan en alumno, un Hash Join entre apoderado y colaborador, y un Index Scan en persona. En 10k datos, se mantiene el mismo enfoque, pero con la adición de Bitmap Heap Scan en colaborador utilizando el nuevo índice. En 100k datos, los costos y el uso de recursos aumentan significativamente, con el uso intensivo de Bitmap Heap Scan en colaborador. Para 1M datos, se introduce la parallelización con Gather y Parallel Hash Semi Join, con Parallel Seq Scan en alumno y matrícula, y Parallel Hash Join entre apoderado y colaborador, además de un Parallel Bitmap Heap Scan en colaborador, demostrando un incremento significativo en costos y eficiencia en el manejo de grandes volúmenes de datos. En todos los casos, se realiza un Index Scan en persona usando el índice primario para buscar los resultados finales.

## 5.4 Plataforma de pruebas

<b>Sistema Operativo</b>	Windows 11 64-bits
<b>RAM</b>	16 GB
<b>CPU</b>	Intel Core i5-1235U
<b>Capacidad SSD</b>	512 GB
<b>PostgreSQL</b>	16.2
<b>DataGrip</b>	2024.1.3
<b>Docker</b>	4.31.1

Table 21: Especificaciones plataforma de pruebas

## 5.5 Medición de tiempos

### 5.5.1 Sin índices

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	175.210	205.306	1459.966	114534.298
2	169.214	216.402	1516.431	117201.023
3	177.710	274.884	1432.166	112984.575
4	164.512	227.705	1627.135	118647.252
5	192.028	250.850	1530.447	110794.072
Promedio	175.7348	235.0294	1513.229	114832.24399
Desviación estándar	9.3607	27.94095	75.31508	75.31508

Table 22: Consulta 1

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	345.052	652.664	44891.185	3931597.627
2	302.450	677.999	45805.199	3701023.021
3	45805.199	734.057	52669.867	3880975.729
4	378.817	692.784	47687.895	3619787.067
5	309.166	710.037	47245.607	3881074.322
Promedio	332.4116	693.5082	47659.9506	3802891.5532
Desviación estándar	30.76274	30.93022	3015.70432	134795.57485

Table 23: Consulta 2

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	167.540	315.738	912.975	58503.063
2	175.244	343.664	988.522	57298.393
3	204.936	279.833	846.328	56484.832
4	185.714	321.168	883.048	56993.090
5	206.292	303.995	909.574	57136.121
Promedio	187.9452	312.8796	908.08940	57283.0998
Desviación estándar	17.37788	23.42840	57.43193	746.90762

Table 24: Consulta 3

### 5.5.2 Con índices por defecto

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	0.251	6.257	396.624	23115.031
2	0.197	5.320	335.431	24299.944
3	0.233	5.137	349.104	25556.929
4	0.172	5.994	341.882	25814.039
5	0.217	5.048	379.886	25312.110
Promedio	0.214	5.5512	360.5854	24819.6106
Desviación estándar	0.03078	0.54139	26.38975	1112.51145

Table 25: Consulta 1

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	0.402	15.265	635.171	18865.481
2	0.514	15.353	657.373	20918.379
3	0.356	0.356	720.721	18851.776
4	0.384	18.379	625.819	20451.714
5	0.483	19.401	677.966	18945.817
Promedio	0.4278	17.214	663.41	19606.6334
Desviación estándar	0.06748	1.84471	37.89260	998.82839

Table 26: Consulta 2

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	0.326	2.826	26.550	124.261
2	0.461	3.347	20.467	119.881
3	0.361	2.632	25.670	112.850
4	0.338	2.287	24.999	116.193
5	0.351	2.979	24.037	125.580
Promedio	0.3674	2.8142	24.3446	119.753
Desviación estándar	0.05396	0.39443	2.35473	5.35297

Table 27: Consulta 3

### 5.5.3 Con índices por defecto más índices personalizados

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	0.161	4.324	82.715	652.607
2	0.167	4.869	88.989	632.291
3	0.153	4.892	86.286	605.283
4	0.150	4.993	70.504	636.185
5	0.159	4.811	81.153	633.012
Promedio	0.158	0.158	81.9294	631.87560
Desviación estándar	0.00670	0.26206	7.08127	17.00729

Table 28: Consulta 1

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	0.279	14.750	590.168	16914.586
2	0.263	16.133	610.265	16854.632
3	0.241	14.500	581.118	17035.101
4	0.306	13.077	570.728	16685.816
5	0.224	14.478	563.422	16854.632
Promedio	0.2626	14.5876	583.1402	16868.95340
Desviación estándar	0.03205	1.08584	18.24900	126.13174

Table 29: Consulta 2

Ejecución	1k (ms)	10k (ms)	100k (ms)	1M (ms)
1	0.269	1.716	23.353	93.977
2	0.252	1.646	22.853	98.822
3	0.239	1.849	17.552	104.173
4	0.385	1.919	17.063	95.269
5	0.262	1.812	18.207	105.109
Promedio	0.2814	1.7884	19.8056	99.47
Desviación estándar	0.05900	0.10817	3.04248	5.05365

Table 30: Consulta 3

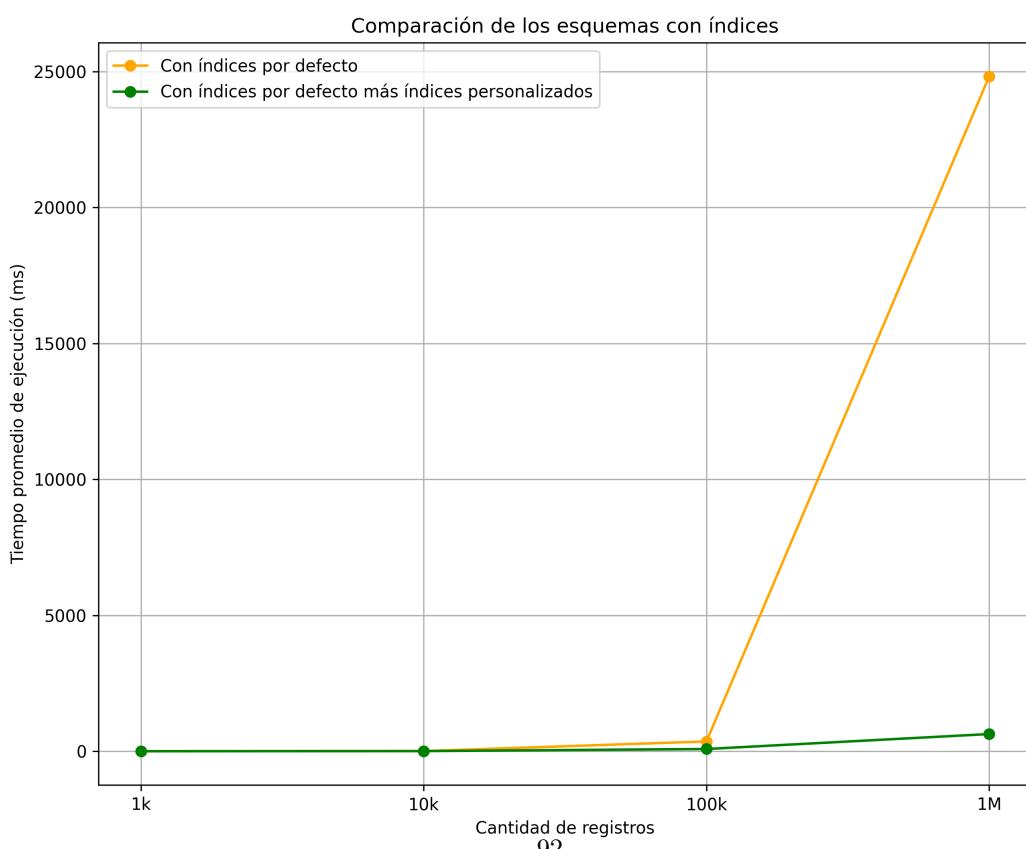
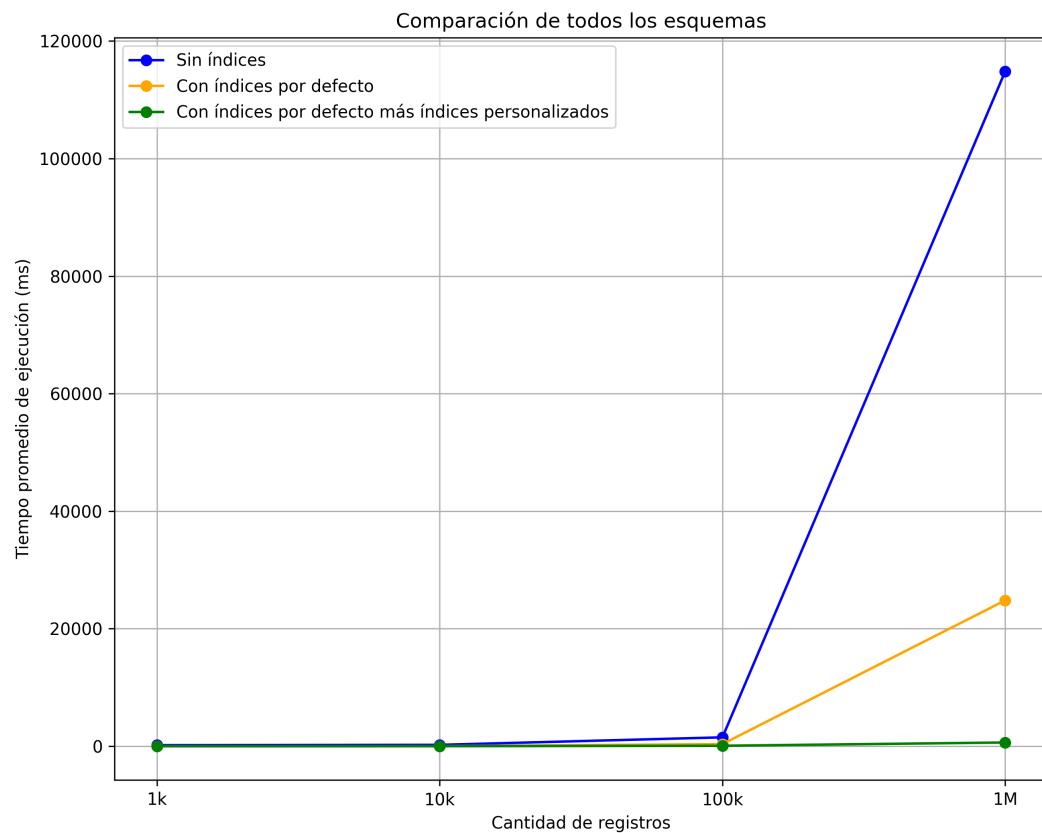
## 5.6 Resultados, análisis y discusión

Debido a la gran escala de los datos, se ha dividido los gráficos en dos partes para facilitar la visualización. Primero mostramos las que incluyen las estadísticas de las consultas sin índices, con índices por defecto y con índices por defecto más índices personalizados. Luego, realizamos la comparación entre estas últimas dos.

Esta división se realizó porque, debido a la escala de las gráficas, la diferencia entre las consultas con índices por defecto y las consultas con índices personalizados no se evidencia en la primera imagen. Sin embargo, dicha diferencia sí existe y es importante resaltarla para un análisis más preciso.



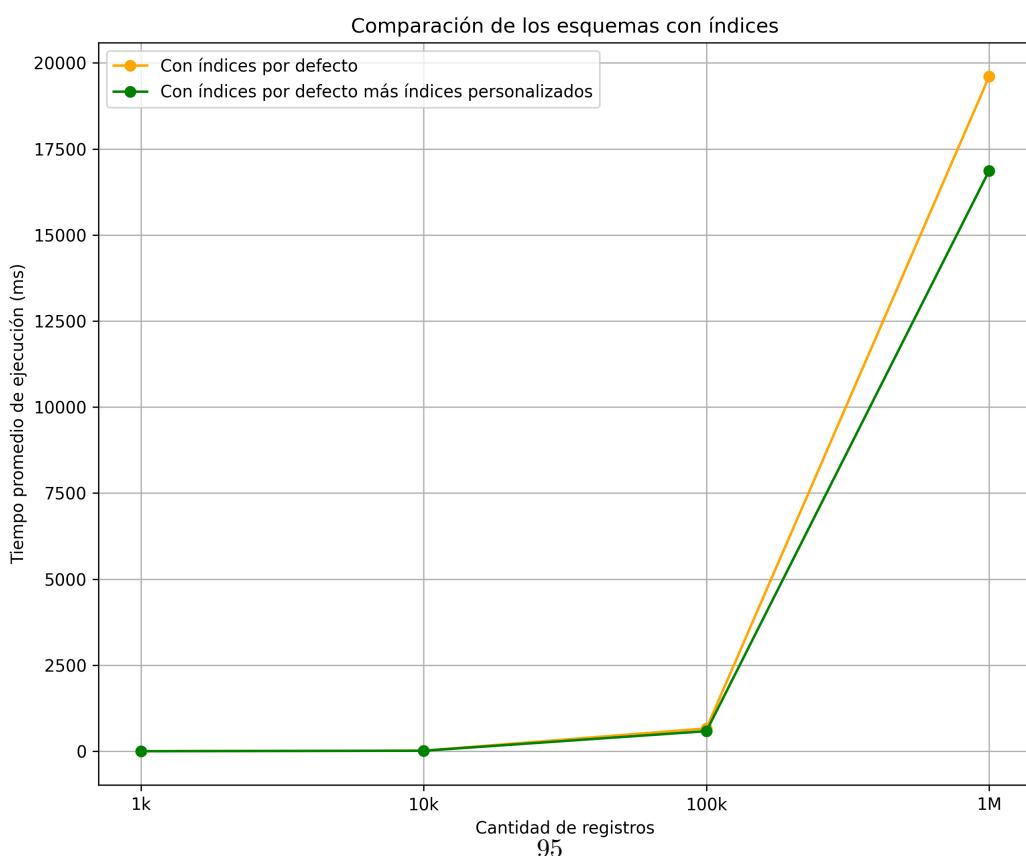
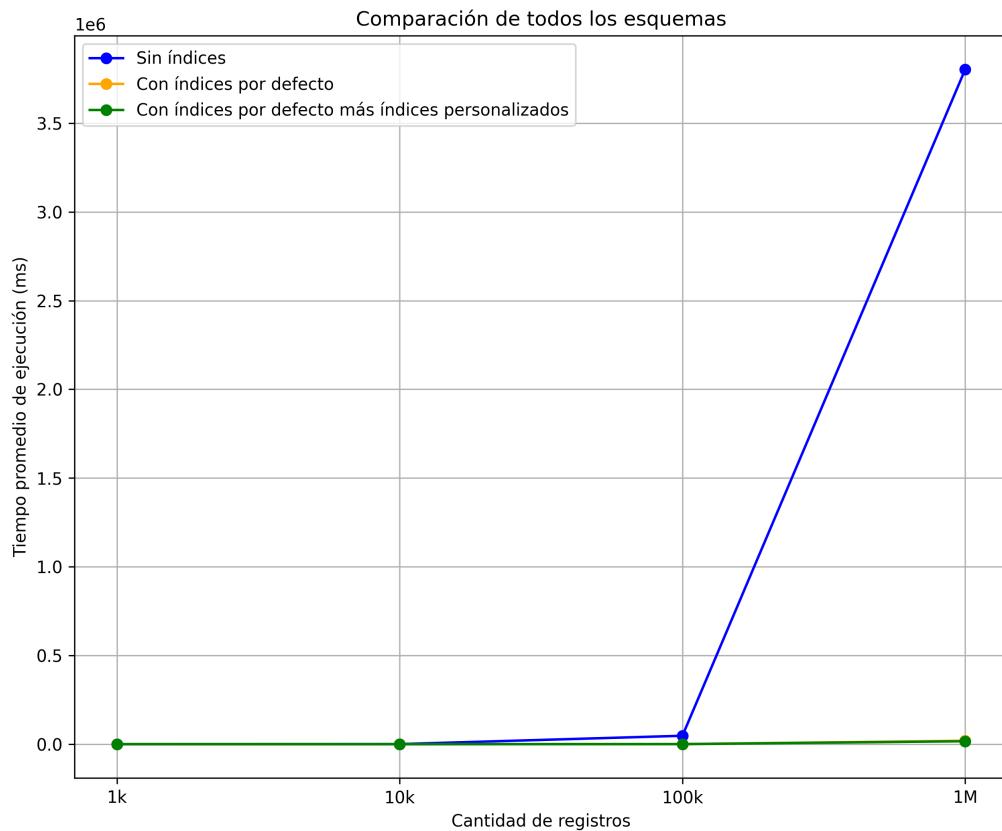
### 5.6.1 Consulta 1



Podemos observar una mejora significativa en los tiempos de ejecución entre las consultas sin índices y las consultas con índices (por defecto y por defecto más personalizado). Esto se debe a que en las consultas sin índices, estas dependen de escaneos secuenciales y uniones anidadas, incrementando exponencialmente el tiempo de ejecución por la gran cantidad de comparaciones necesarias. También identificamos una mejora entre las consultas con índices por defecto y las consultas con índices por defecto más índices personalizados: con índices por defecto, se utilizan escaneos de índice, reduciendo el tiempo de ejecución en un 78.38% para un millón de datos, por otro lado, en el caso de los índices personalizados, la implementación de un índice en sede.construccion\_fecha permite una búsqueda más eficiente y reduce el número de filas examinadas, mejorando el tiempo de ejecución en un 97.45% respecto a las consultas con índices por defecto; esto minimiza los costos de entrada-salida y optimiza las uniones, resultando en tiempos significativamente más rápidos, especialmente para grandes volúmenes de datos. En volúmenes menores, también se observa una mejora notable, aunque no tan pronunciada. Además, la reducción de operaciones de materialización y la mejora en el ordenamiento contribuyen a esta notable mejora en el rendimiento.



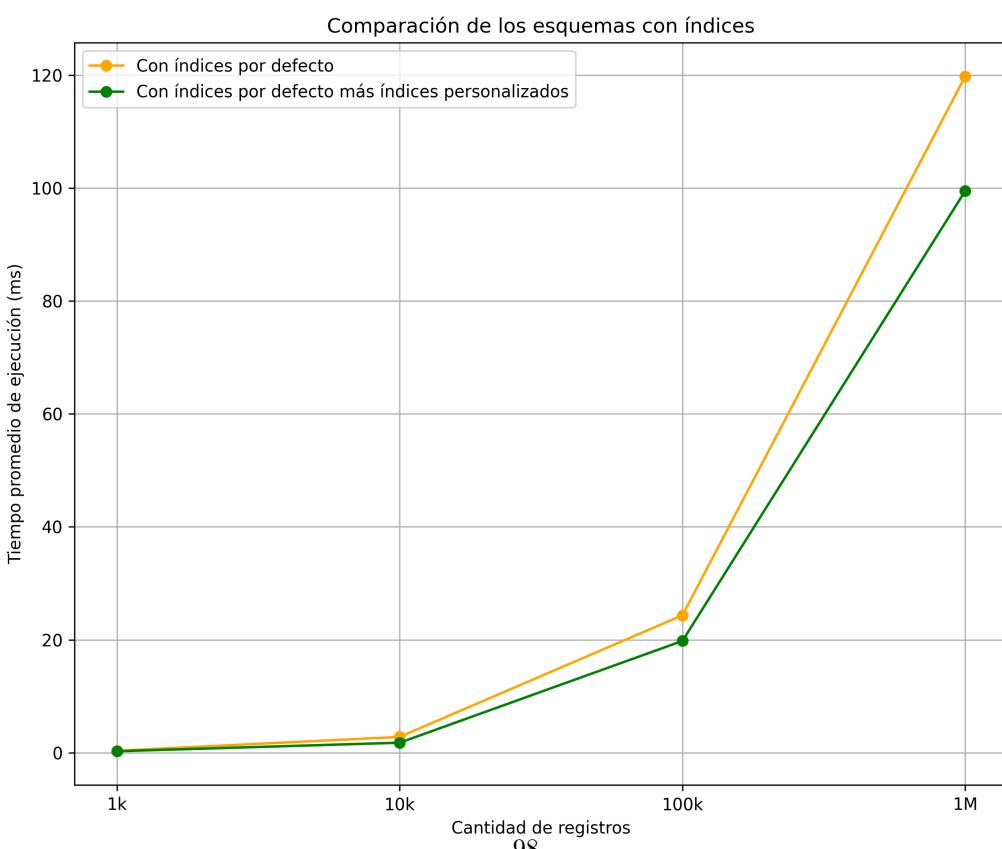
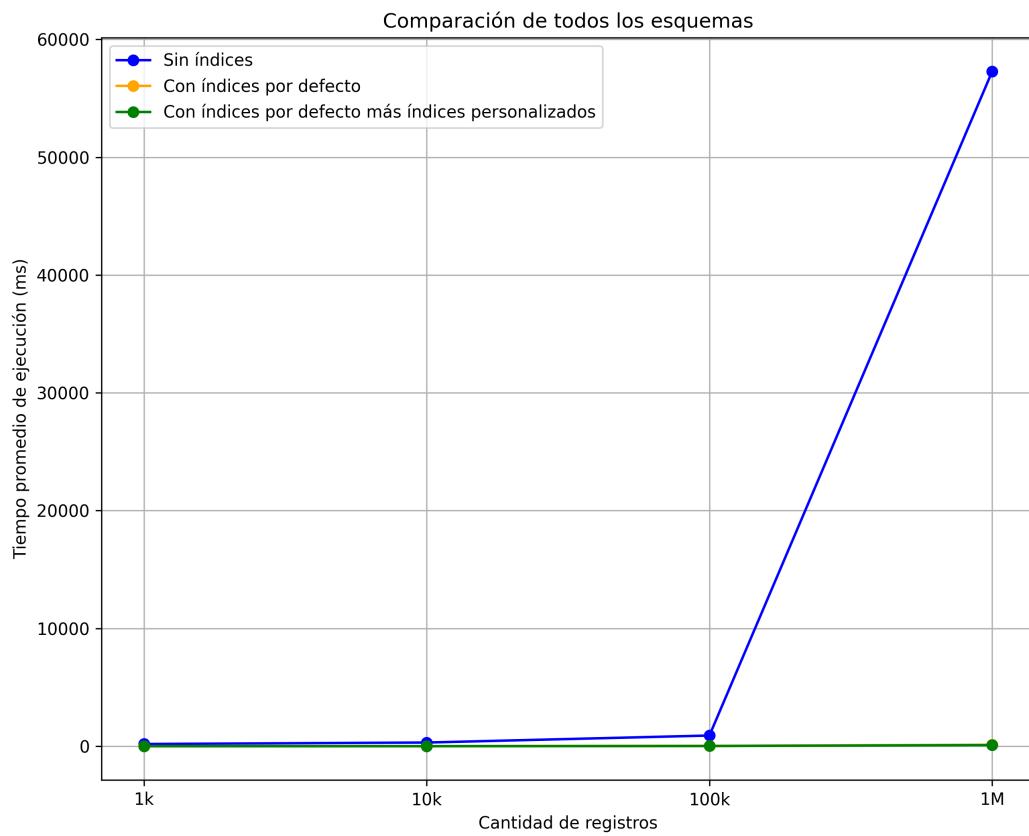
### 5.6.2 Consulta 2



En el caso de la segunda consulta, observamos una evidente mejora en los tiempos de ejecución de las consultas con índices: estos optimizan el acceso y filtrado de datos, siendo estas mejoras más pronunciadas en el esquema de un millón de registros. Sin índices, la consulta realiza escaneos secuenciales y uniones anidadas, resultando en tiempos de ejecución muy elevados debido a la cantidad de comparaciones necesarias. Sin embargo, al activar los índices por defecto, se logró una reducción del 99.50% en el tiempo de ejecución mediante escaneos de índice y uniones hash. Aún así, la mayor eficiencia se obtuvo al implementar un índice en persona.nacimiento\_fecha (más los índices por defecto), disminuyendo el tiempo de ejecución en un 13.91% comparado con las consultas con índices por defecto. Esta optimización se debe a que el índice específico facilita la búsqueda rápida y reduce significativamente las filas a evaluar. Además, el uso de uniones hash y la reducción de operaciones de materialización contribuyeron a este rendimiento mejorado.



### 5.6.3 Consulta 3



Por último, en la tercera consulta —como en las dos anteriores— existen mejoras notables en los tiempos de ejecución de las consultas con índices, especialmente en el escenario de un millón de registros. Sin índices, la consulta depende de escaneos secuenciales y uniones anidadas, lo que resulta en tiempos elevados por la gran cantidad de comparaciones. Al utilizar índices por defecto, logramos una reducción del 99.79% en el tiempo de ejecución gracias a escaneos de índice y uniones hash. La optimización máxima la alcanzamos al implementar un índice en colaborador.horas\_semanales\_trabajo, reduciendo el tiempo de ejecución en un 16.95% adicional respecto a los índices por defecto. Esta mejora se debe a que el índice específico agiliza la búsqueda y disminuye considerablemente las filas a evaluar. Además, el empleo de uniones hash en lugar de anidadas y la menor necesidad de materialización de datos contribuyeron a este rendimiento superior. Aunque en volúmenes menores también se observan mejoras, estas no son tan marcadas como en el caso de 1 millón de registros, donde la efectividad del índice es evidente.

## 6 Conclusiones

A lo largo de este proyecto, hemos llevado a cabo un análisis exhaustivo del rendimiento de la ejecución de las consultas SQL planteadas bajo 3 contextos: sin índices, con índices que el optimizador de PostgreSQL identifica y con índices declarados manualmente junto a los anteriores. Nuestro enfoque se centró en evaluar cómo estos índices o su ausencia afectan el tiempo de ejecución y la eficiencia general de las consultas, especialmente en escenarios con grandes volúmenes de datos. A continuación, presentamos las conclusiones derivadas de nuestro trabajo.

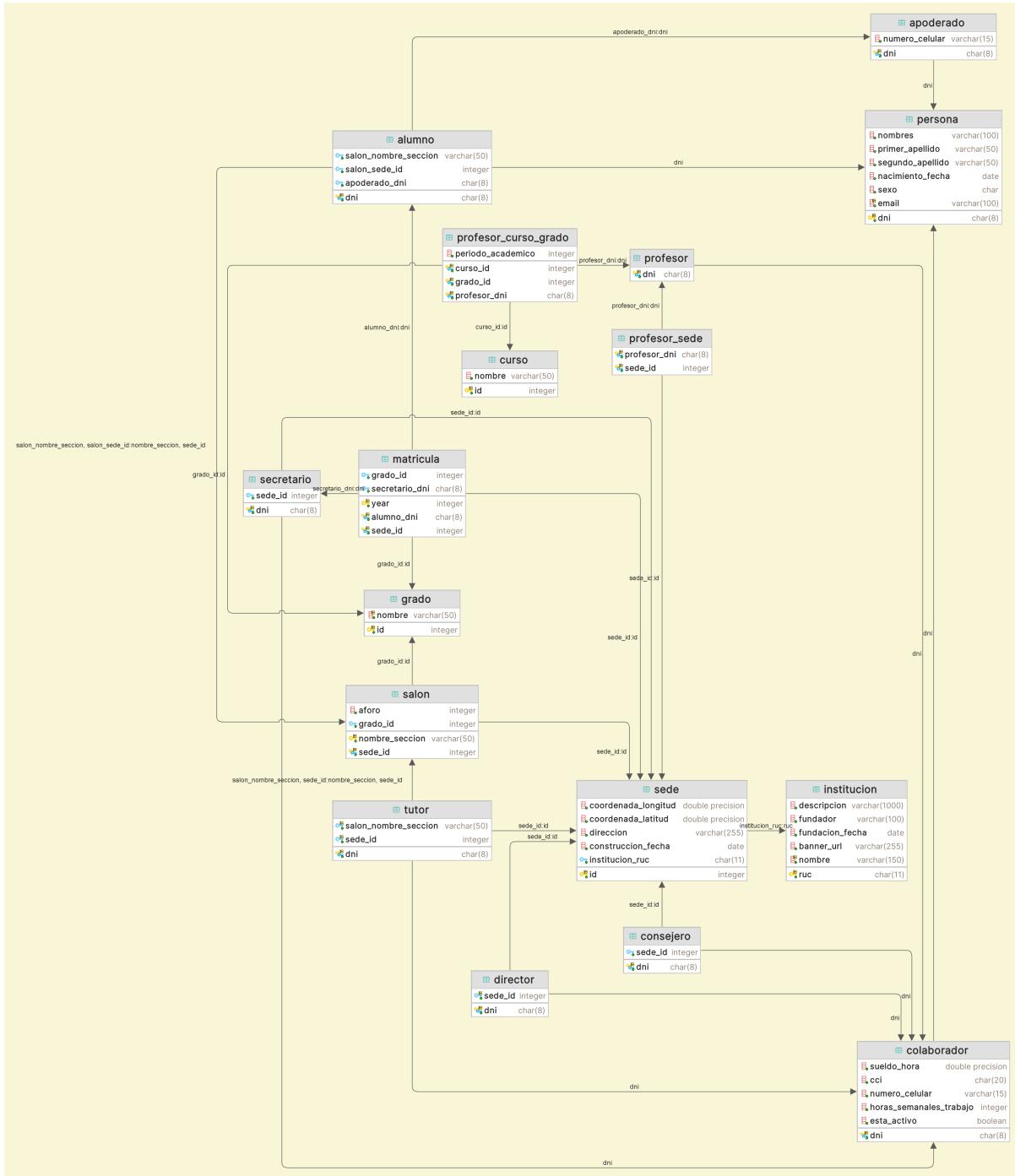
Primero, la implementación de índices personalizados generó mejoras significativas en los tiempos de ejecución de las consultas, siendo estas optimizaciones cruciales en escenarios de un millón de registros, donde la eficiencia del índice se hizo más evidente. Los índices agilizan las búsquedas y disminuyen las filas a evaluar, mejorando considerablemente el rendimiento de la base de datos.

Segundo, el desarrollo del modelo entidad-relación y relacional permitió implementar una base de datos consistente que representa eficazmente el esquema propuesto, facilitando la creación y manipulación eficiente de relaciones entre diferentes entidades.

Por último, las consultas desarrolladas, de alta complejidad, resaltaron la importancia de los planes de consulta y el impacto de los índices en la optimización, demostrando cómo la escalabilidad y la eficiencia se pueden mejorar significativamente mediante una arquitectura de base de datos bien optimizada para manejar grandes volúmenes de datos.

## 7 Anexos

### 7.1 Modelo Físico



### 7.2 Repositorios de GitHub

#### 7.2.1 Repositorio del informe en LaTeX

Este informe se ha realizado en LaTeX con el apoyo de GitHub, aprovechando sus herramientas de control de versiones y colaboración. La plataforma ha permitido documentar y gestionar cada etapa del proyecto con precisión. Para más detalles, acceda al repositorio en el siguiente enlace: [GitHub Repositorio](#).

#### 7.2.2 Repositorio del código en SQL

Los contenidos del proyecto, incluidos el código SQL, el script de Python y los archivos CSV, se encuentran

tran disponibles en otro repositorio de GitHub. Para más detalles, acceda al repositorio en el siguiente enlace: [GitHub Repositorio](#).

### 7.2.3 Repositorio para el front-end

Los contenidos del frontend creado para el proyecto están disponibles en el siguiente enlace: [GitHub Repositorio](#).

## 7.3 Videos de experimentación

### 7.3.1 Consulta 1

- [\*Sin índices\*](#)
- [\*Con índices por defecto\*](#)
- [\*Con índices por defecto más índices personalizados\*](#)

### 7.3.2 Consulta 2

- [\*Sin índices\*](#)
- [\*Con índices por defecto\*](#)
- [\*Con índices por defecto más índices personalizados\*](#)

### 7.3.3 Consulta 3

- [\*Sin índices\*](#)
- [\*Con índices por defecto\*](#)
- [\*Con índices por defecto más índices personalizados\*](#)

## 7.4 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?

Para manejar eficientemente millones de datos, es fundamental optimizar la estructura de índices, particionar tablas, utilizar almacenamiento en caché y considerar bases de datos no relacionales (NoSQL). Sistemas como MongoDB y Cassandra son gestores de bases de datos no relacionales que facilitan la gestión de grandes volúmenes de datos y proporcionan flexibilidad en el modelado de datos. Por otro lado, la arquitectura Cliente-Servidor es suficiente si se optimiza adecuadamente, permite escalabilidad horizontal, implementa平衡adores de carga y mitiga cuellos de botella en la red, CPU y disco. Una arquitectura distribuida puede ser necesaria para gestionar eficientemente las operaciones a gran escala.