

Universidad de Ingeniería y Tecnología

FACULTAD DE COMPUTACIÓN

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

HITO 1

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

PROFESOR:
RIOS OJEDA, BRENNER HUMBERTO

MAYO, 2024

Contenidos

1 Requisitos	3
1.1 Introducción	3
1.2 Descripción general del problema	3
1.3 Necesidad y usos de la base de datos	4
1.4 ¿Cómo resuelve el problema hoy?	4
1.4.1 ¿Cómo se almacenan y procesan los datos?	4
1.4.2 Flujo de datos	4
1.5 Descripción detallada del sistema	5
1.5.1 Objetos de información actuales	5
1.5.2 Características y funcionalidades esperadas	5
1.5.3 Tipos de usuarios existentes o necesarios	5
1.5.4 Tipo de consulta y actualizaciones	5
1.5.5 Tamaño estimado de la base de datos	6
1.6 Objetivos del proyectos	7
1.7 Referencias del proyecto	8
1.8 Eventualidades	8
1.8.1 Problemas que pudieran encontrarse en el proyecto	8
1.8.2 Límites y alcances del proyecto	8
2 Modelo Entidad-Relación	9
2.1 Reglas semánticas	9
2.2 Modelo Entidad-Relación	11
2.3 Especificaciones y consideraciones sobre el modelo	12
2.3.1 Entidad Institucion	12
2.3.2 Entidad Persona	12
2.3.3 Entidad Alumno	12
2.3.4 Entidad Apoderado	12
2.3.5 Entidad Colaborador	12
2.3.6 Entidad Profesor	13
2.3.7 Entidad Secretario	13
2.3.8 Entidad Director	13
2.3.9 Entidad Consejero	13
2.3.10 Entidad Tutor	13
2.3.11 Entidad Sede	14
2.3.12 Entidad Salon	14
2.3.13 Entidad Grado	14
2.3.14 Entidad Curso	14
2.3.15 Entidad Matrícula	14
3 Modelo Relacional	15
3.1 Modelo Relacional	15
3.2 Especificaciones de transformación	16
3.2.1 Entidades	16
3.2.2 Entidades débiles	16

3.2.3	Entidades superclases y subclases	16
3.2.4	Relaciones binarias	17
3.2.5	Relaciones ternarias	17
3.3	Diccionario de datos	17
4	Implementación de la base de datos	24
4.1	Creación de tablas en PostgreSQL	24
4.2	Creación de triggers en PostgreSQL	26
4.3	Carga de datos	32
4.4	Simulación de datos faltantes	32
5	Optimización y experimentación	33
5.1	Consultas SQL para el experimento	33
5.1.1	Descripción del tipo de consultas seleccionadas	33
5.1.2	Implementación de consultas en SQL	34
5.2	Metodología del experimento	36
5.3	Optimización de consultas	37
5.3.1	Planes de índices para la primera consulta	37
5.3.2	Planes de índices para la segunda consulta	46
5.3.3	Planes de índices para la tercera consulta	48
5.4	Plataforma de pruebas	49
5.5	Medición de tiempos	50
5.5.1	Sin índices	50
5.5.2	Con índices por defecto	51
5.5.3	Con índices por defecto más índices personalizados	53
5.6	Resultados, análisis y discusión	55
5.6.1	Consulta 1	55
5.6.2	Consulta 2	55
5.6.3	Consulta 3	55
6	Conclusiones	55
7	Anexos	55
7.1	Modelo Físico	55
7.2	Repositorios de GitHub	56
7.2.1	Repositorio del informe en LaTeX	56
7.2.2	Repositorio del código en SQL	56
7.3	Videos de experimentación	56
7.3.1	Consulta 1	56
7.3.2	Consulta 2	56
7.3.3	Consulta 3	56
7.4	Pregunta extra	56

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 inefficiencias 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 ajenas a esta necesidad, en consecuencia, distintas formas más o menos tecnológicas han sido implementadas para satisfacerla: desde anotaciones a lápiz y papel hasta tablas y tablas de Excel.

1.3 Necesidad y usos de la base de datos

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

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

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

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

1.7 Referencias del proyecto

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

1.8 Eventualidades

1.8.1 Problemas que pudieran encontrarse en el proyecto

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

1.8.2 Límites y alcances del proyecto

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

2 Modelo Entidad-Relación

2.1 Reglas semánticas

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

2.2 Modelo Entidad-Relación

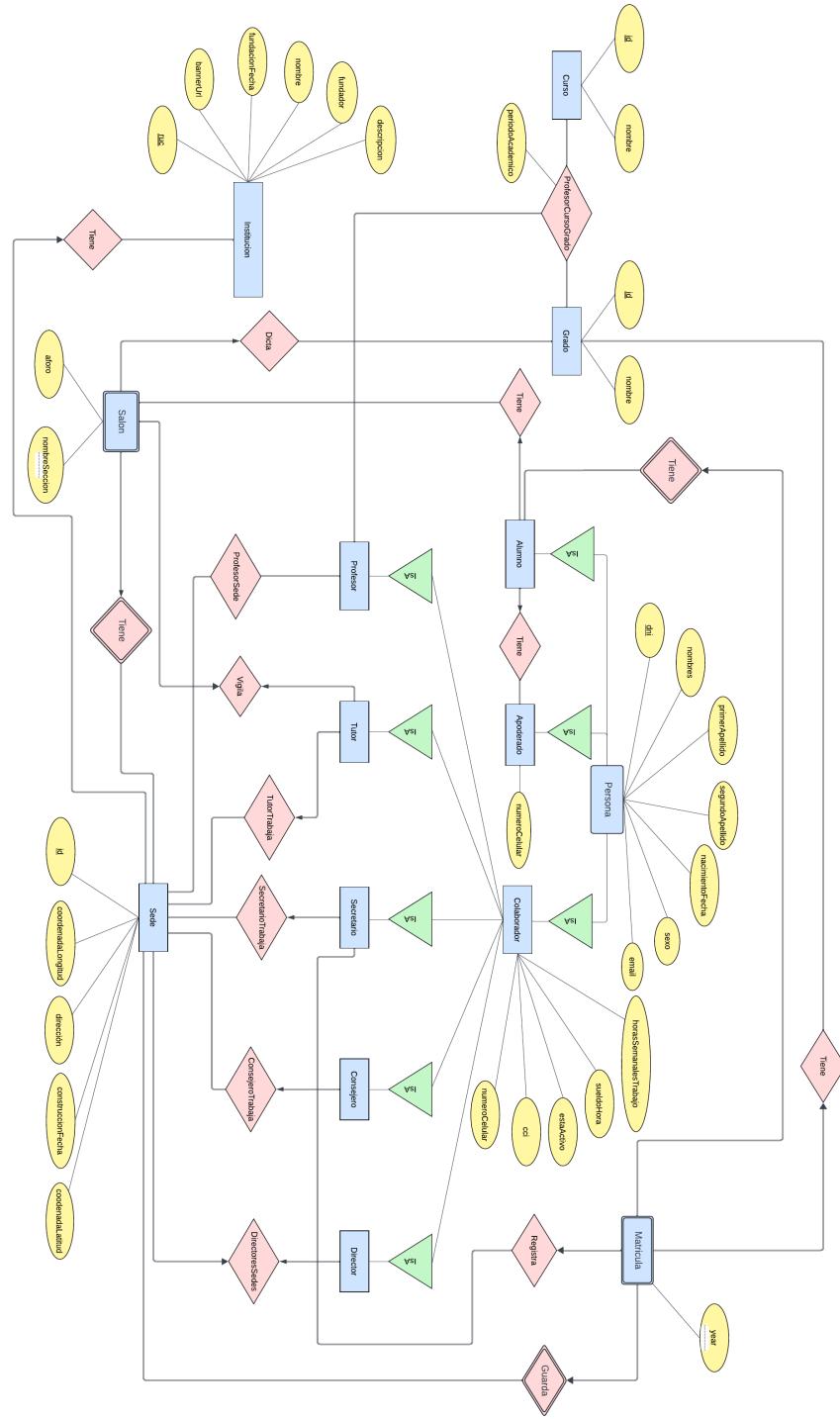


Figure 1: Modelo Entidad-Relación

2.3 Especificaciones y consideraciones sobre el modelo

2.3.1 Entidad Institucion

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

2.3.2 Entidad Persona

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

2.3.3 Entidad Alumno

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

2.3.4 Entidad Apoderado

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

2.3.5 Entidad Colaborador

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

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

2.3.6 Entidad Profesor

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

2.3.7 Entidad Secretario

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

2.3.8 Entidad Director

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

2.3.9 Entidad Consejero

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

2.3.10 Entidad Tutor

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

2.3.11 Entidad Sede

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

2.3.12 Entidad Salon

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

2.3.13 Entidad Grado

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

2.3.14 Entidad Curso

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

2.3.15 Entidad Matrícula

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

3 Modelo Relacional

3.1 Modelo Relacional

- Institucion (ruc, descripcion, fundador, fundacionFecha, bannerUrl, nombre)
- Persona (dni, nombres, primerApellido, segundoApellido, nacimientoFecha, sexo, email)
- Apoderado (Persona.dni, numeroCelular)
- Alumno (Persona.dni, Salon.nombreSeccion, Sede.id, Apoderado.dni)
- Colaborador (Persona.dni, sueldoHora, cci, numeroCelular, horasSemanalesTrabajo, estaActivo)
- Secretario (Colaborador.dni, Sede.id)
- Consejero (Colaborador.dni, Sede.id)
- Director (Colaborador.dni, Sede.id)
 - Relación One To One (1:1) con la entidad Sede, usamos el constraint UNIQUE en el traspaso hacia las tablas SQL, para asegurar que un Director solo pueda ser asignado a una sede.
- Tutor (Colaborador.dni, Salon.nombreSeccion, Sede.id)
 - Relación One To One (1:1) con la entidad Salon, usamos el constraint UNIQUE en el traspaso hacia las tablas SQL, para asegurar que un Tutor solo pueda ser asignado a un salon.
- Profesor (Colaborador.dni)
- ProfesorSede (Profesor.dni, Sede.id)
- Sede (id, coordenadaLongitud, coordenadaLatitud, direccion, construcionFecha, 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 construcionFecha. Representa una ubicación física donde se imparten cursos y trabajan los colaboradores.
- **Institucion:** Se convierte en la tabla Institucion con ruc como llave primaria. Incluye descripcion, fundador, fundacionFecha, bannerUrl, nombre. Esta entidad encapsula los datos fundamentales de la institución educativa.

3.2.2 Entidades débiles

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

3.2.3 Entidades superclases y subclases

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

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

3.2.4 Relaciones binarias

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

3.2.5 Relaciones ternarias

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

3.3 Diccionario de datos

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

Table 4: Alumno

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

Table 5: Apoderado

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

Table 6: Colaborador

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

Table 7: Consejero

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

Table 8: Curso

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

Table 9: Director

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

Table 10: Grado

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

Table 11: Institucion

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

Table 12: Matricula

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

Table 13: Persona

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

Table 14: ProfesorCursoGrado

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

Table 15: ProfesorSede

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

Table 16: Profesor

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

Table 17: Salon

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

Table 18: Secretario

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

Table 19: Sede

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

Table 20: Tutor

4 Implementación de la base de datos

4.1 Creación de tablas en PostgreSQL

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

```

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

```

```

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

```

4.2 Creación de triggers en PostgreSQL

1 CREATE OR REPLACE FUNCTION

```

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

```

```

      tabla hija.', new.dni;
51   END IF;

52
53   RETURN new;
54 END;
55 $$ LANGUAGE plpgsql;

56
57 CREATE TRIGGER trigger_check_profesor_overlapping
58   BEFORE INSERT
59   ON profesor
60   FOR EACH ROW
61 EXECUTE FUNCTION function_check_colaborador_overlapping();

62
63 CREATE TRIGGER trigger_check_consejero_overlapping
64   BEFORE INSERT
65   ON consejero
66   FOR EACH ROW
67 EXECUTE FUNCTION function_check_colaborador_overlapping();

68
69 CREATE TRIGGER trigger_check_secretario_overlapping
70   BEFORE INSERT
71   ON secretario
72   FOR EACH ROW
73 EXECUTE FUNCTION function_check_colaborador_overlapping();

74
75 CREATE TRIGGER trigger_check_director_overlapping
76   BEFORE INSERT
77   ON director
78   FOR EACH ROW
79 EXECUTE FUNCTION function_check_colaborador_overlapping();

80
81 CREATE TRIGGER trigger_check_tutor_overlapping
82   BEFORE INSERT
83   ON tutor
84   FOR EACH ROW
85 EXECUTE FUNCTION function_check_colaborador_overlapping();

86
87 -----
88
89 CREATE OR REPLACE FUNCTION
90   function_check_colaborador_apoderado_overlapping_alumno()
91   RETURNS TRIGGER AS
92   $$
93 BEGIN
94   IF EXISTS (SELECT 1 FROM alumno WHERE dni = new.dni) THEN
95     RAISE EXCEPTION 'La persona con DNI % ya existe en la tabla
96     Alumno.', new.dni;
97   END IF;

98   RETURN new;
99 END;
100 $$ LANGUAGE plpgsql;

101
102 CREATE TRIGGER trigger_check_colaborador_overlapping
103   BEFORE INSERT
104   ON colaborador
105   FOR EACH ROW

```

```

104 EXECUTE FUNCTION
105     function_check_colaborador_apoderado_overlapping_alumno();
106 CREATE TRIGGER trigger_check_apoderado_overlapping
107     BEFORE INSERT
108     ON apoderado
109     FOR EACH ROW
110 EXECUTE FUNCTION
111     function_check_colaborador_apoderado_overlapping_alumno();
112 -----
113
114 CREATE OR REPLACE FUNCTION function_check_colaborador_esta_activo()
115     RETURNS TRIGGER AS
116 $$
117 DECLARE
118     esta_activo BOOLEAN;
119 BEGIN
120     SELECT esta_activo
121     INTO esta_activo
122     FROM colaborador
123     WHERE new.dni = dni;
124
125     IF NOT esta_activo THEN
126         RAISE EXCEPTION 'El colaborador que se intenta insertar no
127         esta activo.';
128     END IF;
129
130     RETURN new;
131 END;
132 $$ LANGUAGE plpgsql;
133
134 CREATE TRIGGER trigger_check_profesor_esta_activo
135     BEFORE INSERT OR UPDATE
136     ON persona
137     FOR EACH ROW
138 EXECUTE FUNCTION function_check_colaborador_esta_activo();
139 -----
140
141 CREATE OR REPLACE FUNCTION gestionar_director_reasignacion()
142     RETURNS TRIGGER AS
143 $$
144 BEGIN
145     IF (SELECT COUNT(*) FROM director WHERE sede_id = old.sede_id
146     AND dni != old.dni) = 0 THEN
147         RAISE EXCEPTION 'No se puede reasignar el director sin
148         reemplazo en la sede %', old.sedeid;
149     END IF;
150
151     DELETE FROM director WHERE dni = old.dni;
152
153     UPDATE colaborador SET esta_activo = FALSE WHERE dni = old.dni;
154
155     RETURN new;
156 END;
157 $$ LANGUAGE plpgsql;

```

```

154
155 CREATE TRIGGER trigger_gestionar_director_reasignacion
156   AFTER UPDATE
157   ON director
158   FOR EACH ROW
159   WHEN (old.dni IS DISTINCT FROM new.dni)
160 EXECUTE FUNCTION gestionar_director_reasignacion();
161
162 -----
163
164 CREATE OR REPLACE FUNCTION function_check_matricula_year() RETURNS
165   TRIGGER AS
166   $$$
167 BEGIN
168   IF new.year <= (SELECT EXTRACT(YEAR FROM construccion_fecha)
169     FROM sede WHERE id = new.sede_id) THEN
170     RAISE EXCEPTION 'El año de matrícula debe ser mayor que el
171       año de construcción de la sede.';
172   END IF;
173   RETURN new;
174 END;
175 $$ LANGUAGE plpgsql;
176
177
178 CREATE TRIGGER trigger_check_matricula_year
179   BEFORE INSERT OR UPDATE
180   ON matricula
181   FOR EACH ROW
182 EXECUTE FUNCTION function_check_matricula_year();
183
184 -----
185
186 CREATE OR REPLACE FUNCTION function_check_salon_aforo_on_alumno()
187   RETURNS TRIGGER AS
188   $$$
189 DECLARE
190   salon_aforo      INT;
191   alumnos_cantidad INT;
192
193 BEGIN
194   SELECT aforo
195     INTO salon_aforo
196     FROM salon
197    WHERE nombre_sección = new.nombre_sección
198      AND sede_id = new.sede_id;
199
200   SELECT COUNT(dni) + 1
201     INTO alumnos_cantidad
202     FROM alumno
203    WHERE salon_nombre_sección = new.salon_nombre_sección
204      AND salon_sede_id = new.salon_sede_id;
205
206   IF alumnos_cantidad > salon_aforo THEN
207     RAISE EXCEPTION 'El aforo del salón ha sido excedido. Aforo
208       máximo: %, Número de alumnos: %', salon_aforo,
209       alumnos_cantidad;
210   END IF;
211
212   RETURN new;

```

```

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

```

```

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

```

4.3 Carga de datos

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

4.4 Simulación de datos faltantes

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

5 Optimización y experimentación

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

5.1 Consultas SQL para el experimento

5.1.1 Descripción del tipo de consultas seleccionadas

- **Consulta 1:** La siguiente consulta obtiene el nombre, email y número de celular de los directores actuales, así como la dirección, coordenadas y fecha de construcción de las 20 sedes más antiguas construidas entre 1990 y 2010, excluyendo aquellas sedes donde el número total de profesores y alumnos 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 sus colaboradores que nacieron entre 1960 y 1980.
 - **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 mayores que trabajan en esa sede. Para obtener el monto de bonificación se debe cuadruplicar la multiplicación de la remuneración establecida por hora por la cantidad de horas semanales laboradas. De existir un colaborador que labore en más de una sede se debe evaluar la condición del aniversario para todas las sedes; si se cumple la

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

- **Consulta 3:** Se desea conocer el nombre, primer apellido, segundo apellido, sexo e email de los alumnos menores de 18 años cuyo apoderado sea un colaborador registrado como activo, que trabaja a tiempo completo (más de 48 horas semanales) y cuyo sueldo mensual sea menor a 2000 soles (el sueldo mensual se obtiene cuadruplicando la multiplicación del pago por hora por las horas semanales). Además deben haber transcurrido como mínimo 2 años desde la matrícula del alumno.
 - **Justificacion:** 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 || ' '
3         ,|| d_persona.segundo_apellido AS nombre_director,
4     d_persona.email AS email_director,
5     d_colaborador.numero_celular AS celular_director,
6     sede.direccion,
7     sede.coordenada_longitud,
8     sede.coordenada_latitud,
9     sede.construccion_fecha
10 FROM
11     director
12         JOIN colaborador AS d_colaborador ON director.dni =
13     d_colaborador.dni
14         JOIN persona AS d_persona ON d_colaborador.dni =
15     d_persona.dni
16         JOIN sede ON director.sede_id = sede.id
17 WHERE
18     sede.construccion_fecha BETWEEN '1990-01-01' AND ,
19     2010-12-31,
20 AND (
21     SELECT COUNT(*)
22     FROM profesor_sede
23     WHERE profesor_sede.sede_id = sede.id
24 ) + (
25     SELECT COUNT(*)
26     FROM alumno
27     WHERE alumno.salon_sede_id = sede.id
```

```

24      ) <= 400
25 ORDER BY
26     sede.construccion_fecha
27 LIMIT 20;

```

- Consulta 2:

```

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

```

- Consulta 3:

```

1 SELECT persona.nombres ,
2      persona.primer_apellido ,
3      persona.segundo_apellido ,
4      persona.sexo ,
5      persona.email
6 FROM persona
7 WHERE persona.dni
8     IN (SELECT alumno.dni
9          FROM alumno
10         WHERE alumno.dni
11             IN (SELECT matricula.alumno_dni
12                  FROM matricula
13                  WHERE matricula.year <= EXTRACT(YEAR FROM
14                      CURRENT_DATE) - 2)
15                  AND alumno.apoderado_dni
16                      IN (SELECT apoderado.dni
17                            FROM apoderado
18                            WHERE apoderado.dni
19                                IN (SELECT colaborador.dni
20                                      FROM colaborador
21                                      WHERE colaborador.está_activo
22                                          = TRUE
23                                          AND colaborador.
24                                              horas_semanales_trabajo > 48
25                                              AND colaborador.sueldo_hora *
26                                              colaborador.horas_semanales_trabajo * 4 < 2000)))
27 AND persona.nacimiento_fecha > CURRENT_DATE - INTERVAL '18
28 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.

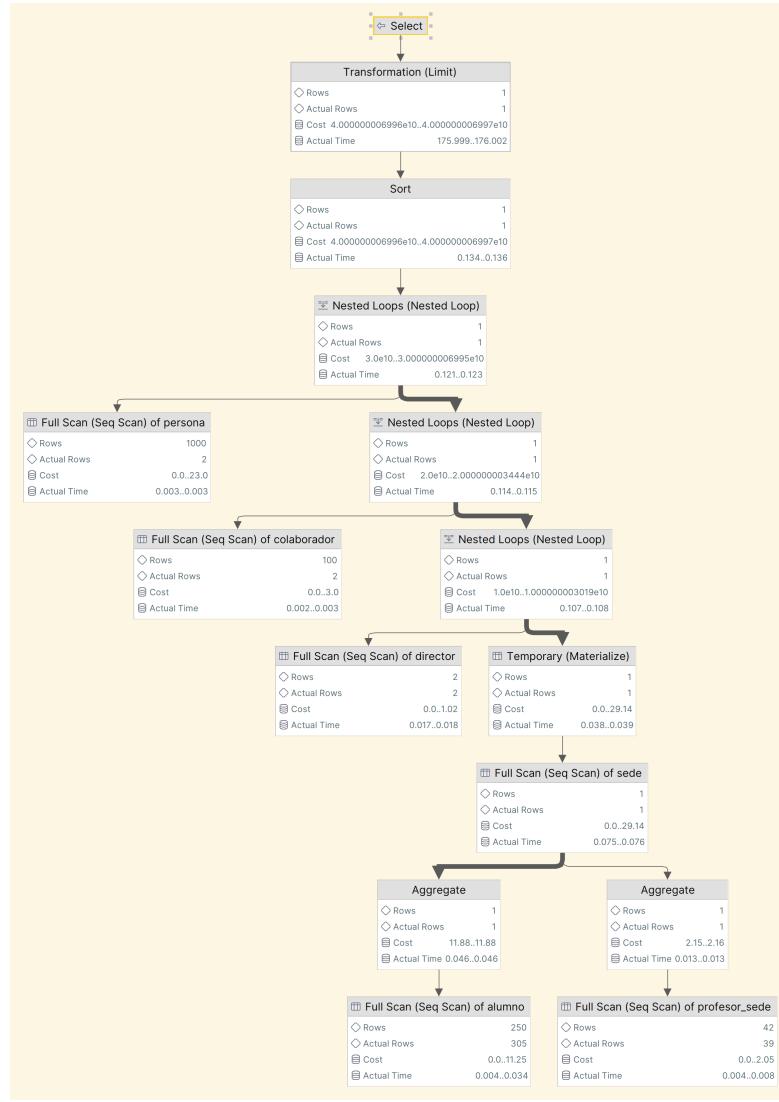
5.3 Optimización de consultas

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:



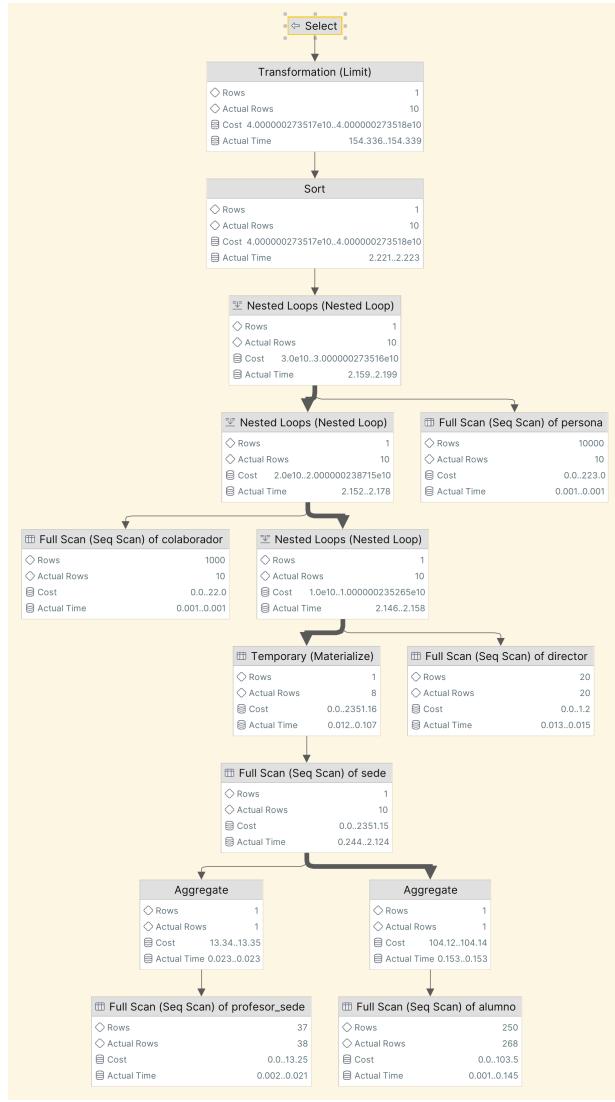
```

Limit (cost=4088000000.96 .. 4800000000.97 rows=1 width=600) (actual time=100.248..100.259 rows=1 loops=1)
-> Sort (cost=4000000000.96 .. 4800000000.97 rows=1 width=600) (actual time=0.120..0.122 rows=1 loops=1)
  Sort Key: sede.construccion.fecha
  Sort Method: quicksort  Memory: 258B
-> Nested Loop (cost=3000000000.00 .. 3080000000.95 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=2000000000.00 .. 2080000000.44 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=1000000000.00 .. 1080000000.19 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 .. 0.02 rows=2 width=40) (actual time=0.012..0.013 rows=2 loops=1)
-> Materialize (cost=0.00 .. 0.14 rows=1 width=540) (actual time=0.037..0.037 rows=1 loops=2)
-> Seq Scan on sede (cost=0.00 .. 0.14 rows=1 width=540) (actual time=0.072..0.073 rows=1 loops=1)
  Filter: ((construccion.fecha >= '1928-01-01'::date) AND (construccion.fecha <= '2018-12-31'::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 .. 0.05 rows=42 width=8) (actual time=0.004..0.034 rows=365 loops=1)
  Filter: (salon.sede_id = sede.id)
  Rows Removed by Filter: 45
-> Seq Scan on sede (cost=0.00 .. 0.02 rows=1 width=8) (actual time=0.002..0.002 rows=1 loops=1)
-> Seq Scan on profesor_sede (cost=0.00 .. 0.02 rows=1 width=8) (actual time=0.002..0.002 rows=1 loops=1)
  Filter: (sede_id = sede.id)
  Rows Removed by Filter: 195
-> Seq Scan on colaborador_d_colaborador (cost=0.00 .. 0.00 rows=108 width=19) (actual time=0.002..0.002 rows=2 loops=1)
-> Seq Scan on persona_d_persona (cost=0.00 .. 0.00 rows=1088 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 1.679 ms, Inlining 14.923 ns, Optimization 107.488 ms, Emission 67.725 ms, Total 191.884 ms
Execution Time: 192.020 ms

```

- Para diez mil datos:



```

Limit (cost=40000303105.09..40000303105.14 rows=28 width=129) (actual time=1528.882..1528.890 rows=28 loops=1)
-> Sort (cost=40000303105.09..40000303105.18 rows=37 width=129) (actual time=1271.565..1271.571 rows=28 loops=1)
  Sort Key: sede.construccion.fecha
  Sort Method: top-N heapsort Memory: 33KB
-> Nested Loop (cost=300000000000.00..300000303164.12 rows=37 width=129) (actual time=2.167..1271.521 rows=187 loops=1)
  Join Filter: (director.dni = d_persona.dni)
  Rows Removed by Join Filter: 10009993
-> Seq Scan on persona d_persona (cost=0.00..2226.00 rows=180000 width=68) (actual time=0.821..8.915 rows=100000 loops=1)
-> Materialize (cost=20000000000.00..2000024543.84 rows=37 width=92) (actual time=0.880..0.887 rows=107 loops=180000)
  -> Nested Loop (cost=20000000000.00..2000024543.66 rows=37 width=92) (actual time=2.139..329.937 rows=107 loops=1)
    Join Filter: (director.dni = d_colaborador.dni)
    Rows Removed by Join Filter: 10690993
-> Seq Scan on colaborador d_colaborador (cost=0.00..214.00 rows=18000 width=19) (actual time=0.002..0.699 rows=18000 loops=1)
-> Materialize (cost=10000000000.00..10000239673.75 rows=37 width=73) (actual time=0.080..0.628 rows=107 loops=18000)
  -> Nested Loop (cost=10000000000.00..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: 15622
-> 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=20000000000.00..2000024543.18 rows=37 width=68) (actual time=0.011..1.215 rows=79 loops=200)
  -> Seq Scan on sede (cost=0.00..239559.18 rows=37 width=68) (actual time=2.121..241.820 rows=187 loops=1)
    Filter: ((construccion.fecha >= '1920-01-01'::date) AND (construccion.fecha <= '2018-12-31'::date) AND (((SubPlan 1) + (SubPlan 2)) <= 187))
    Rows Removed by Filter: 93
  SubPlan 1
    -> Aggregate (cost=105.12..105.12 rows=1 width=8) (actual time=0.322..0.323 rows=1 loops=111)
      -> Seq Scan on profesor_sede (cost=0.00..165.00 rows=46 width=8) (actual time=0.088..0.317 rows=46 loops=111)
        Filter: (sede_id = sede.id)
        Rows Removed by Filter: 9154
  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 19.621 ms, Optimization 138.416 ms, Emission 99.908 ms, Total 258.767 ms
Execution Time: 1530.447 ms

```

- Para cien mil datos:



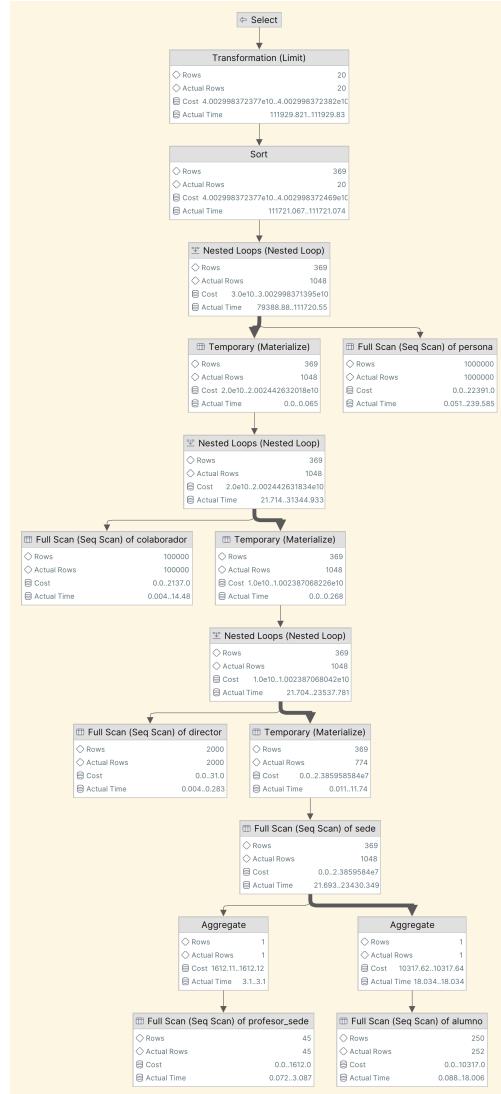
```

Limit (cost=40880002735.17..480000002735.18 rows=1 width=600) (actual time=249.378..249.378 rows=10 loops=1)
-> Sort (cost=40000002735.17..480000002735.18 rows=1 width=600) (actual time=2.841..2.844 rows=10 loops=1)
  Sort Key: sede.construccion.fecha
  Sort Method: quicksort  Memory: 278B
-> Nested Loop (cost=300000000000.00..308000002735.16 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..208000002387.15 rows=1 width=591) (actual time=2.773..2.883 rows=10 loops=1)
  Join Filter: (director.dni = d_colaborador.dni)
  Rows Removed by Join Filter: 94
-> Nested Loop (cost=100000000000.88..108000002352.65 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: 10
-> Seq Scan on director (cost=0.00..1.38 rows=28 width=40) (actual time=0.813..0.815 rows=28 loops=1)
-> Materialize (cost=0.00..2351.16 rows=1 width=540) (actual time=0.812..0.817 rows=0 loops=28)
-> Seq Scan on sede (cost=0.00..2351.15 rows=1 width=540) (actual time=0.241..2.737 rows=10 loops=1)
  Filter: ((construccion.fecha >= '1928-01-01'::date) AND (construccion.fecha <= '2018-12-31'::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=8) (actual time=0.002..0.026 rows=38 loops=12)
  Filter: (sede_id = sede.id)
  Rows Removed by Filter: 702
-> Aggregate (cost=184.12..184.14 rows=1 width=8) (actual time=0.197..0.197 rows=1 loops=12)
-> Seq Scan on alumno (cost=0.00..103.58 rows=258 width=8) (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=3880 width=39) (actual time=0.001..0.001 rows=10 loops=10)
-> Seq Scan on persona_d_persona (cost=0.00..223.00 rows=18800 width=59) (actual time=0.001..0.001 rows=18 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 28.686 ms, Optimization 144.891 ms, Emission 80.962 ms, Total 247.876 ms
Execution Time: 250.850 ms

```

- Para un millón de datos:



```

Limit (cost=40029983723.77..40029983723.82 rows=28 width=131) (actual time=110759.067..110759.876 rows=20 loops=1)
-> Sort (cost=100000000000.00..100000000000.00 rows=369 width=131) (actual time=110502.459..110502.467 rows=20 loops=1)
  Sort Key: sede.construcion_fecha
  Sort Method: top-N heapsort Memory: 344B
-> Nested Loop (cost=300000000000.00..30029983713.95 rows=369 width=131) (actual time=77752.126..110502.064 rows=1048 loops=1)
  Join Filter: (director.dni = d.persona.dni)
  Rows Removed by Join Filter: 104799952
-> Seq Scan on persona d_persona (cost=0.00..22391.00 rows=1000000 width=61) (actual time=0.881..216.767 rows=1000000 loops=1)
-> Materialize (cost=20000000000.00..20024425320.18 rows=369 width=93) (actual time=0.880..0.883 rows=1048 loops=1)
  -> Nested Loop (cost=20000000000.00..20024425318.34 rows=369 width=93) (actual time=28.843..38135.045 rows=1048 loops=1)
    Join Filter: (director.dni = d.colaborador.dni)
    Rows Removed by Join Filter: 104799952
-> Seq Scan on colaborador d_colaborador (cost=0.00..2127.00 rows=1000000 width=19) (actual time=0.094..13.891 rows=1000000 loops=1)
-> Materialize (cost=10000000000.00..10023870662.26 rows=369 width=74) (actual time=0.089..0.255 rows=1048 loops=1000000)
  -> Nested Loop (cost=10000000000.00..10023870668.42 rows=369 width=74) (actual time=29.831..22215.573 rows=1048 loops=1)
    Join Filter: (director.sede_id = sede.id)
    Rows Removed by Join Filter: 104799952
-> Seq Scan on director (cost=0.00..31.00 rows=2800 width=13) (actual time=0.004..0.259 rows=2800 loops=1)
-> Materialize (cost=0.00..23859585.84 rows=369 width=69) (actual time=0.018..11.879 rows=774 loops=2800)
  -> Seq Scan on sede (cost=0.00..23859584.00 rows=369 width=69) (actual time=20.818..22118.626 rows=1048 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: 952
  SubPlan 1
    -> Aggregate (cost=1012.11..1012.12 rows=1 width=8) (actual time=2.945..2.945 rows=1 loops=1108)
      -> Seq Scan on profesor_sede (cost=0.00..1012.00 rows=45 width=8) (actual time=0.069..2.934 rows=45 loops=1108)
        Filter: (sede_id = sede.id)
        Rows Removed by Filter: 89955
  SubPlan 2
    -> Aggregate (cost=1013.62..1013.64 rows=1 width=8) (actual time=16.997..16.997 rows=1 loops=1108)
      -> Seq Scan on alumno (cost=0.00..1013.58 rows=250 width=0) (actual time=0.081..16.972 rows=252 loops=1)
        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 0.936 ms, Inlining 42.138 ms, Optimization 115.887 ms, Emission 98.653 ms, Total 257.689 ms
Execution Time: 110794.872 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.construcion_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.construcion_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

- 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 escaño 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

- 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 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 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

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

5.6.1 Consulta 1

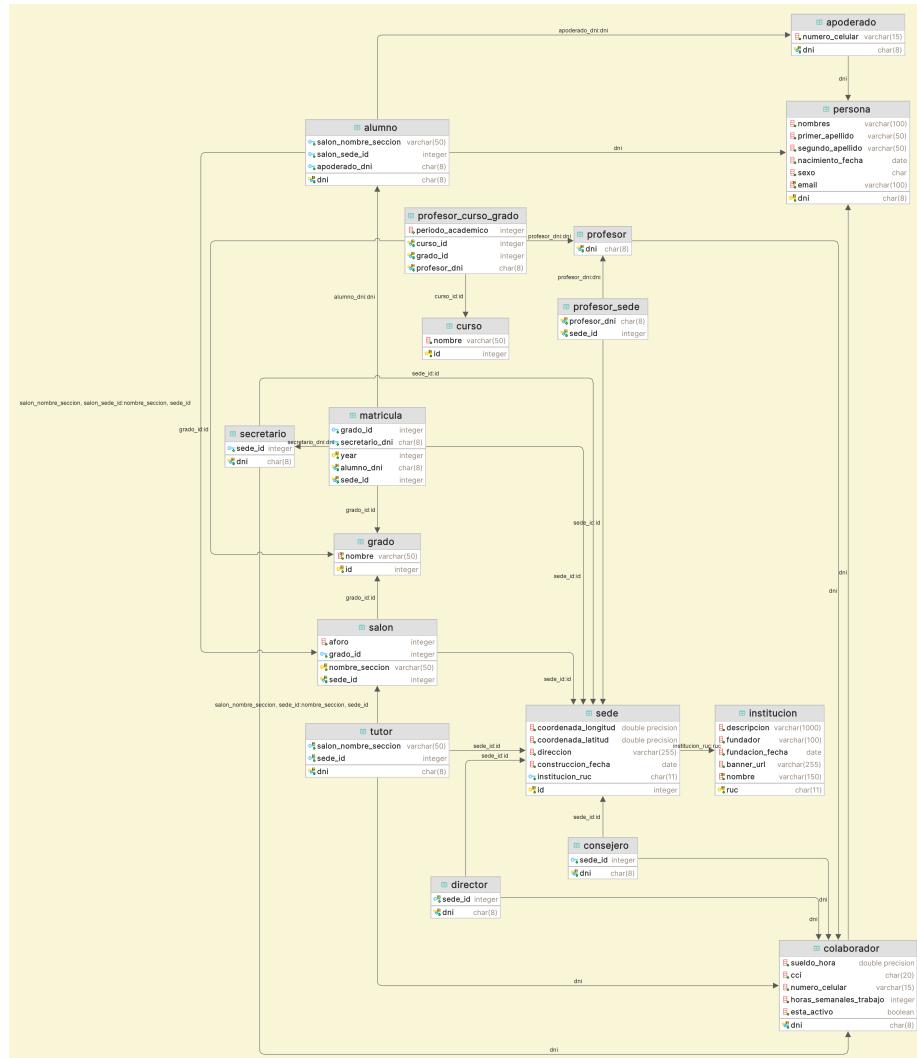
5.6.2 Consulta 2

5.6.3 Consulta 3

6 Conclusiones

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 disponibles en otro repositorio de GitHub. Para más detalles, acceda al repositorio 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 balanceadores 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.