UNIVERSIDAD DE LOS LAGOS

PROYECTO SISTEMA PARA UNIVERSIDAD

PROYECTO MULTIASIGNATURA

DEPARTAMENTO DE CIENCIAS DE LA INGENIERÍA
INGENIERÍA CIVIL EN INFORMÁTICA
ASIGNATURA
CAMPUS OSORNO, CHILE

Autor
Autor

8 de febrero de 2026

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Capítulo 1

Formulación del Proyecto

1.1. Introducción

El presente documento se ha elaborado en respuesta a la solicitud de [Nombre del Cliente o Empresa Cliente] para el desarrollo de un sistema de base de datos web destinado a gestionar la información académica y de investigación de su institución. Comprendemos la necesidad de contar con una plataforma robusta, escalable y eficiente que permita administrar de manera integrada los datos relativos a profesores, alumnos, proyectos, departamentos y asignaturas.

Este informe tiene como objetivo principal presentar una propuesta detallada del modelo de datos que servirá como pilar fundamental para la construcción del sistema. Un modelo de datos bien definido es crucial para garantizar la integridad de la información, facilitar el desarrollo de funcionalidades complejas, optimizar las consultas y asegurar la futura adaptabilidad del sistema a nuevas necesidades.

La información contenida en las siguientes secciones ha sido cuidadosamente analizada a partir de los requisitos proporcionados. Se expondrá la identificación de las entidades principales, sus atributos y las relaciones que existen entre ellas. Esta estructura ha sido pensada para ser comprensible tanto por el equipo de gerencia, para la toma de decisiones estratégicas, como por el equipo de desarrollo, quienes se encargarán de la implementación técnica.

Confiamos en que este análisis y la propuesta de diseño que se presenta a continuación constituirán una base sólida para el exitoso desarrollo del sistema de base de datos web que su institución requiere. Estamos a su disposición para discutir cualquier aspecto de este informe y para colaborar estrechamente en las siguientes fases del proyecto.

Perfecto. A continuación, te presento una propuesta para el objetivo general y los objetivos específicos del proyecto, considerando las etapas que mencionaste:

1.2. Objetivos del Proyecto

1.2.1. Objetivo General

Desarrollar e implementar un sistema de base de datos web integral y eficiente que satisfaga los requisitos de gestión de información académica y de investigación de [Nombre del Cliente o Empresa Cliente], permitiendo la administración centralizada y estructurada de datos relativos a profesores, alumnos, proyectos, departamentos y asignaturas, y facilitando el acceso y la operatividad a los usuarios designados.

1.2.2. Objetivos Específicos

1. **Formulación y Modelado Conceptual**: Analizar exhaustivamente los requisitos funcionales y de datos proporcionados por el cliente, para formular el presente informe técnico y diseñar un modelo de datos conceptual y lógico que represente fielmente las entidades, atributos y relaciones identificadas, sirviendo como base para la estructura de la base de datos.

2. **Diseño e Implementación de la Base de Datos**: Traducir el modelo de datos lógico a un diseño físico de base de datos optimizado, seleccionando un sistema gestor de base de datos (SGBD) adecuado, y proceder con la creación de la estructura de la base de datos, incluyendo tablas, relaciones, restricciones de integridad y los índices necesarios para asegurar la eficiencia y consistencia de los datos.

3. **Diseño y Desarrollo del Sistema Web con Integración de Datos**: Diseñar la arquitectura del sistema web y desarrollar una interfaz de usuario intuitiva y funcional que permita la interacción con la base de datos implementada. Esto incluye la implementación de los procesos CRUD (Crear, Leer, Actualizar, Eliminar) para todas las entidades gestionadas, asegurando una correcta integración entre la capa de presentación y la capa de datos.

4. **Pruebas, Validación y Despliegue**: Ejecutar un plan de pruebas exhaustivo que abarque pruebas unitarias, de integración, de sistema y de aceptación del usuario, con el fin de validar el correcto funcionamiento del sistema web y la base de datos, asegurar el cumplimiento de los requisitos iniciales, corregir posibles errores y preparar el sistema para su despliegue en el entorno productivo del cliente.

1.3. Metodología de Trabajo

A continuación, se presenta el desglose de actividades y tareas necesarias para la consecución de cada objetivo específico del proyecto.

1.3.1. Objetivo Específico 1: Formulación y Modelado Conceptual

Actividades y Tareas

1. Elaboración del Informe Técnico y Planificación Temporal del Informe
   - 1.1 Revisión y validación final de la comprensión de los requisitos del cliente.
   - 1.2 Definición detallada de la estructura y secciones del presente informe (Índice).
   - 1.3 Redacción de la Introducción y Justificación del proyecto (ya avanzado).
   - 1.4 Redacción de los Objetivos General y Específicos (ya avanzado).
   - 1.5 Desarrollo de la sección de Planificación de Actividades y Tareas (en curso).
   - 1.6 Estimación de tiempos para cada sección del informe y elaboración de un cronograma para la entrega del mismo.
   - 1.7 Consolidación y revisión general del informe técnico.
2. Análisis de Requisitos y Modelado Entidad-Relación (MER)
   - 2.1 Identificación de Entidades: Extraer y listar todas las entidades principales a partir de los requisitos del cliente (Profesor, Proyecto, Alumno, ProgramaPostgrado, Departamento, Asignatura).
   - 2.2 Definición de Atributos: Para cada entidad, detallar sus atributos, especificando el tipo de dato preliminar y las restricciones conocidas (ej. RUN como identificador único).
     - 2.2.1 Profesor: RUN (PK), nombre, edad, rango, especialidad_investigacion.
     - 2.2.2 Proyecto: numero_proyecto (PK), nombre_patrocinador, fecha_comienzo, fecha_finalizacion, presupuesto.
     - 2.2.3 Alumno: RUN (PK), nombre, edad.
     - 2.2.4 ProgramaPostgrado: codigo_programa (PK), nombre_programa.
     - 2.2.5 Departamento: numero_departamento (PK), nombre_departamento, despacho_principal.
     - 2.2.6 Asignatura: (Se necesitará un identificador, ej. codigo_asignatura (PK), nombre_asignatura).
   - 2.3 Identificación de Relaciones: Determinar las relaciones entre las entidades basándose en los requisitos.
     - 2.3.1 Profesor dirige Proyecto (Investigador Principal).
     - 2.3.2 Profesor trabaja en Proyecto (Investigador).
     - 2.3.3 Alumno Postgrado trabaja en Proyecto (Ayudante).
     - 2.3.4 Profesor supervisa Alumno en Proyecto.
     - 2.3.5 Alumno pertenece (o no) a ProgramaPostgrado.
     - 2.3.6 Departamento tiene un Profesor Director.
     - 2.3.7 Profesor trabaja en Departamento (con % de tiempo).
     - 2.3.8 Alumno Postgrado tiene un Departamento Principal.

2.3.9 Alumno Postgrado tiene un Alumno Asesor.

2.3.10 Profesor dicta Asignatura.

2.4 Especificación de Cardinalidad y Opcionalidad: Definir la cardinalidad (uno a uno, uno a muchos, muchos a muchos) y la opcionalidad (obligatoria o opcional) para cada relación.

2.5 Creación del Diagrama Entidad-Relación (DER): Elaborar el diagrama visual utilizando una notación estándar (Chen, Crow's Foot, etc.) que represente las entidades, atributos y relaciones con su cardinalidad.

2.6 Documentación y Validación del MER: Describir textualmente el MER, justificando las decisiones de diseño y validarlo contra los requisitos para asegurar que toda la información y reglas de negocio estén correctamente representadas.

1.3.2. Objetivo Específico 2: Diseño e Implementación de la Base de Datos

Actividades y Tareas

1. Transformación del Modelo Entidad-Relación a Modelo Relacional

   1.1 Mapear cada entidad del MER a una tabla en el modelo relacional.

   1.2 Mapear los atributos de las entidades a columnas en sus respectivas tablas, definiendo tipos de datos SQL específicos.

   1.3 Convertir las relaciones del MER:

      1.3.1 Relaciones 1:N se implementan mediante claves foráneas en la tabla del lado "N".

      1.3.2 Relaciones M:N se convierten en una nueva tabla asociativa con claves foráneas a las tablas originales.

      1.3.3 Relaciones 1:1 se analizan para decidir la ubicación de la clave foránea o si se fusionan las tablas.

   1.4 Definir formalmente las Claves Primarias (PK) para cada tabla y las Claves Foráneas (FK) para implementar las relaciones.

2. Normalización de la Base de Datos

   2.1 Analizar las dependencias funcionales en cada tabla propuesta.

   2.2 Verificar y aplicar la Primera Forma Normal (1FN): Asegurar que todos los atributos sean atómicos y no haya grupos repetitivos.

   2.3 Verificar y aplicar la Segunda Forma Normal (2FN): Asegurar que la tabla esté en 1FN y que todos los atributos no clave dependan completamente de la clave primaria completa (relevante para claves primarias compuestas).

2.4 Verificar y aplicar la Tercera Forma Normal (3FN): Asegurar que la tabla esté en 2FN y que no existan dependencias transitivas (atributos no clave que dependan de otros atributos no clave).

2.5 Documentar el proceso de normalización y las decisiones tomadas para cada tabla.

3. Creación de Scripts de Creación de Tablas (DDL) con Restricciones de Integridad

3.1 Escribir las sentencias SQL CREATE TABLE para cada tabla definida en el modelo relacional normalizado.

3.2 Incluir la definición de columnas con sus tipos de datos SQL precisos.

3.3 Especificar las restricciones de PRIMARY KEY para cada tabla.

3.4 Especificar las restricciones de FOREIGN KEY con las cláusulas REFERENCES y acciones referenciales (ej. ON DELETE CASCADE, ON UPDATE RESTRICT).

3.5 Añadir otras restricciones de integridad necesarias: NOT NULL, UNIQUE, CHECK (ej. para rangos de valores, formatos específicos como el RUN, etc.).

3.6 Organizar los scripts en un orden lógico que respete las dependencias referenciales para su correcta ejecución.

4. Carga de Datos de Prueba y Verificación de Consistencia

4.1 Diseñar un conjunto de datos de prueba coherentes y representativos que cubran diversas casuísticas (ej. profesores con y sin proyectos, alumnos en distintos programas, proyectos con múltiples investigadores y ayudantes).

4.2 Escribir sentencias SQL INSERT INTO para poblar las tablas con los datos de prueba.

4.3 Ejecutar los scripts de carga de datos.

4.4 Verificar la consistencia de los datos cargados mediante consultas SELECT, comprobando que las relaciones y restricciones se cumplen (ej. un alumno de postgrado tiene un departamento principal válido, un proyecto tiene un investigador principal existente).

5. Implementación de Índices y Definición de Consultas y Vistas Predefinidas

5.1 Identificar columnas frecuentemente usadas en cláusulas WHERE, JOIN, ORDER BY que podrían beneficiarse de la creación de índices.

5.2 Crear índices (CREATE INDEX) sobre dichas columnas para optimizar el rendimiento de las consultas.

5.3 Diseñar y escribir consultas SQL complejas que el sistema podría requerir frecuentemente (ej. Listar todos los proyectos de un profesor, obtener los ayudantes de un proyecto específico con sus supervisores, mostrar profesores por departamento con su porcentaje de tiempo).

5.4 Crear vistas (CREATE VIEW) para simplificar el acceso a datos que requieren uniones complejas o para presentar subconjuntos específicos de datos a ciertos roles de usuario.

5.5 Evaluar la capacidad de respuesta y fiabilidad del sistema ejecutando las consultas y vistas predefinidas con los datos de prueba, analizando sus planes de ejecución si es necesario.

1.3.3. Objetivo Específico 3: Diseño y Desarrollo del Sistema Web con Integración de Datos

Actividades y Tareas

1. Configuración del Entorno de Desarrollo Web

   1.1 Seleccionar e instalar un servidor web (ej. Apache, Nginx). Instalar y configurar el intérprete de PHP y las extensiones necesarias (ej. para la conexión con la base de datos). Asegurar que el entorno de desarrollo tenga acceso al servidor de base de datos.

2. Diseño de Maquetas del Sistema (Wireframes y Mockups)

   2.1 Crear wireframes de bajo nivel para esbozar la estructura y navegación de las principales pantallas del sistema web (ej. login, listados, formularios de alta/edición).

   2.2 Desarrollar mockups de mayor fidelidad que muestren la apariencia visual (colores, tipografía, disposición de elementos) para la interfaz de usuario general, la interfaz de usuario limitado y la interfaz de administrador.

3. Desarrollo del Frontend (HTML, CSS)

   3.1 Traducir los mockups a código HTML5 semántico para estructurar las páginas web.

   3.2 Desarrollar hojas de estilo CSS3 para aplicar el diseño visual, la tipografía, colores y el layout definido en los mockups.

   3.3 Crear estilos CSS diferenciados para la interfaz de usuario con funcionalidades limitadas y la interfaz de administrador, asegurando una apariencia distintiva para cada una.

   3.4 Considerar principios de diseño responsivo básico para la correcta visualización en diferentes tamaños de pantalla.

4. Desarrollo del Backend (PHP) y Conexión a la Base de Datos

   4.1 Establecer la lógica de conexión segura desde PHP a la base de datos implementada (ej. usando PDO o MySQLi para MySQL, pg_connect para PostgreSQL, etc.).

4.2 Organizar el código PHP en una estructura modular y mantenible (ej. separando lógica de presentación, lógica de negocio y acceso a datos).

5. Implementación de Lógica de Negocio, Procesos CRUD y Automatización (PHP, Funciones/Triggers PL/SQL o similar)

5.1 Desarrollar scripts PHP para manejar las solicitudes HTTP (GET, POST) y ejecutar la lógica de negocio.

5.2 Implementar funciones PHP para realizar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para cada entidad gestionable a través del sistema web.

5.3 Realizar validaciones de datos tanto en el lado del cliente (JavaScript, opcional) como, fundamentalmente, en el lado del servidor (PHP) antes de interactuar con la base de datos.

5.4 Diseñar e implementar funciones y/o procedimientos almacenados en el SGBD (ej. en PL/pgSQL para PostgreSQL, T-SQL para SQL Server, PL/SQL para Oracle) para encapsular lógica de negocio compleja, mejorar la seguridad o el rendimiento.

5.5 Diseñar e implementar triggers en la base de datos para automatizar procesos (ej. actualizar una fecha de modificación, mantener logs de auditoría básicos, verificar ciertas condiciones antes de una inserción/actualización) y asegurar la integridad referencial o de negocio que no se pueda cubrir con restricciones declarativas.

6. Diseño e Implementación de Interfaces de Usuario Específicas por Rol

6.1 Desarrollar la interfaz de usuario “estándar” o “limitada”, enfocada en las necesidades de usuarios con permisos restringidos, ofreciendo las funcionalidades CRUD que les correspondan (ej. un alumno podría ver sus datos y proyectos, pero no modificar datos de profesores).

6.2 Desarrollar la interfaz de “administrador” con acceso completo a todas las funcionalidades CRUD del sistema, gestión de usuarios (si aplica), y posiblemente otras herramientas de mantenimiento o configuración.

6.3 Implementar un sistema de autenticación y autorización para controlar el acceso a las diferentes interfaces y funcionalidades según el rol del usuario.

6.4 Asegurar que cada interfaz tenga la apariencia distintiva definida con CSS.

1.3.4. Objetivo Específico 4: Pruebas, Validación y Despliegue

Actividades y Tareas

1. Establecimiento del Plan de Pruebas y Validación

1.1 Definir la estrategia general de pruebas: tipos de pruebas a realizar, alcance, criterios de aceptación.

1.2 Diseñar casos de prueba detallados para cada funcionalidad del sistema, cubriendo flujos exitosos, casos límite y manejo de errores.

1.3 Especificar los datos de entrada y los resultados esperados para cada caso de prueba.

1.4 Preparar el entorno de pruebas, asegurando que sea lo más similar posible al entorno de producción.

2. Ejecución de Pruebas Unitarias y de Integración

2.1 Probar individualmente los módulos de código PHP, funciones PL/SQL (o similar) y otros componentes para verificar su correcto funcionamiento aislado (pruebas unitarias).

2.2 Probar la interacción entre los componentes del sistema: frontend con backend, backend con la base de datos, y la correcta ejecución de funciones/triggers en la BD al ser invocados por la aplicación (pruebas de integración).

3. Ejecución de Pruebas de Sistema y Funcionales

3.1 Realizar pruebas del sistema completo para verificar que todas las funcionalidades implementadas operan según los requisitos especificados.

3.2 Probar la correcta implementación de las operaciones CRUD para todas las entidades y roles de usuario.

3.3 Verificar la lógica de negocio, las reglas de validación y la integridad de los datos.

3.4 Evaluar la usabilidad y navegabilidad de las interfaces de usuario (general, limitada y administrador).

3.5 Realizar pruebas de seguridad básicas para identificar vulnerabilidades comunes (ej. Cross-Site Scripting, Inyección SQL, manejo de sesiones).

4. Pruebas de Aceptación del Usuario (UAT)

4.1 Coordinar y facilitar sesiones de prueba con el cliente o usuarios finales designados.

4.2 Guiar a los usuarios en la ejecución de los casos de prueba definidos o en la exploración libre del sistema.

4.3 Recopilar y documentar el feedback de los usuarios, incluyendo errores encontrados, sugerencias de mejora y confirmación de funcionalidades.

5. Corrección de Errores y Refinamiento del Sistema

5.1 Analizar los errores y problemas reportados durante todas las fases de prueba.

5.2 Priorizar y asignar la corrección de errores al equipo de desarrollo.

5.3 Implementar las correcciones necesarias en el código y/o en la base de datos.

5.4 Realizar pruebas de regresión para asegurar que las correcciones no hayan introducido nuevos problemas.

5.5 Refinar la interfaz de usuario y las funcionalidades basándose en el feedback recibido, si se considera pertinente y dentro del alcance.

6. Preparación para el Despliegue (Actividades Preliminares)

6.1 Elaborar la documentación final del sistema: manual de usuario (dirigido a los distintos roles), manual técnico (para mantenimiento y futuras ampliaciones).

6.2 Preparar los scripts y procedimientos necesarios para el despliegue del sistema en un entorno de producción.

6.3 Planificar la capacitación de los usuarios finales, si es necesario.

1.4. Equipo de Desarrollo del Proyecto y Asignación de Responsabilidades

1. Javier Rojas

- **Rol Principal:** Jefe de Proyecto & Desarrollador Full-Stack Senior
- **Responsabilidades Clave:**
  - Coordinación general del equipo y del proyecto.
  - Planificación de sprints/iteraciones y seguimiento del cronograma (Viabilidad de Plazos).
  - Principal punto de comunicación con el “cliente” (profesor/ayudante del curso).
  - Gestión de riesgos y resolución de impedimentos.
  - Supervisión técnica general y toma de decisiones arquitectónicas clave.
  - Desarrollo de componentes backend críticos y lógica de negocio compleja (Objetivo 3).
  - Supervisión y validación del modelado de datos (Objetivo 1).
  - Revisión de la optimización de la base de datos (Objetivo 2).
  - Participación en pruebas unitarias y de integración (Objetivo 4).
  - Coordinación de las Pruebas de Aceptación del Usuario (UAT) y del plan de despliegue (Objetivo 4).

2. Carolina Silva

- **Rol Principal:** Analista de Sistemas & Diseñadora de Bases de Datos

- **Responsabilidades Clave:**
  - Liderazgo en el Objetivo 1:
    - Elaboración de la sección técnica del informe inicial.
    - Análisis detallado de los requisitos del cliente.
    - Creación y documentación del Modelo Entidad-Relación (MER).
  - Liderazgo en el diseño de la BD (parte del Objetivo 2):
    - Transformación del MER al Modelo Relacional.
    - Proceso de Normalización de la base de datos (hasta 3FN).
    - Especificación detallada de tablas, atributos, claves primarias y foráneas, y restricciones de integridad para los scripts DDL.
  - Colaboración en la validación de maquetas desde la perspectiva del flujo de datos (Objetivo 3).
  - Participación activa en la documentación técnica inicial.

3. Martín Pizarro

- **Rol Principal:** Desarrollador Backend & Administrador de Base de Datos (DBA) Junior
- **Responsabilidades Clave:**
  - Liderazgo en la implementación de la BD (parte del Objetivo 2):
    - Escritura de los scripts DDL (`CREATE TABLE`) basados en el diseño de Carolina.
    - Carga de datos de prueba y verificación de consistencia inicial.
    - Implementación de índices, consultas y vistas predefinidas.
  - Liderazgo en el desarrollo backend (parte del Objetivo 3):
    - Configuración de la conexión PHP a la base de datos.
    - Desarrollo de la lógica de negocio principal y funciones PHP para las operaciones CRUD.
    - Implementación de funciones y/o triggers (PL/SQL o similar, según SGBD) para automatización y asegurar integridad en la BD.
    - Optimización de consultas a la base de datos.
  - Colaboración en la configuración del entorno de desarrollo web.
  - Participación en pruebas unitarias y de integración del backend.

4. Sofía Castro

- **Rol Principal:** Desarrolladora Full-Stack & Diseñadora UI/UX
- **Responsabilidades Clave:**

- Liderazgo en el diseño y desarrollo frontend (Objetivo 3):
  - Diseño de maquetas del sistema (wireframes y mockups) para las interfaces de usuario y administrador.
  - Desarrollo de la estructura HTML5 y estilos CSS3, asegurando la apariencia distintiva para cada rol.
  - Implementación de la interfaz de usuario, enfocándose en la usabilidad y experiencia del usuario.
- Colaboración en la definición de requisitos desde la perspectiva del usuario (Objetivo 1).
- Desarrollo de componentes PHP para la integración del frontend con el backend y la lógica CRUD.
- Participación activa en las pruebas funcionales y de usabilidad (Objetivo 4).
- Soporte durante las Pruebas de Aceptación del Usuario (UAT), especialmente en aspectos de interfaz.

5. Diego Valdés

- **Rol Principal:** Desarrollador Full-Stack & Soporte de Entornos
- **Responsabilidades Clave:**
  - Liderazgo en la configuración del entorno de desarrollo y pruebas (Objetivo 3):
    - Montaje del servidor web, PHP y SGBD local o en la nube.
  - Soporte en la carga de datos de prueba y verificaciones de consistencia (Objetivo 2).
  - Desarrollo de módulos frontend y backend según se requiera, apoyando a Martín y Sofía (Objetivo 3):
    - Implementación de funcionalidades CRUD y lógica de negocio.
  - Participación activa en pruebas unitarias y de integración.
  - Elaboración de parte del manual técnico y guías de despliegue (Objetivo 4).
  - Apoyo en la corrección de errores detectados durante las pruebas.

6. Valentina Herrera

- **Rol Principal:** Encargada de Calidad (QA) & Documentación Técnica
- **Responsabilidades Clave:**
  - Liderazgo en el Objetivo 4 (Pruebas y Validación):
    - Establecimiento del plan de pruebas y diseño de casos de prueba.

    - Ejecución de pruebas funcionales, de sistema y de regresión.
    - Registro y seguimiento de errores (bug tracking).
    - Coordinación de las pruebas de integración desde la perspectiva de QA.
    - Preparación del entorno y guía durante las Pruebas de Aceptación del Usuario (UAT).
- Gestión de la Documentación del Proyecto:
    - Consolidación y mantenimiento de la documentación técnica (MER, diseño de BD, manuales).
    - Elaboración del manual de usuario.
    - Revisión y corrección de estilo de la documentación generada por el equipo (incluido el informe inicial).
- Apoyo en la redacción y estructuración del informe inicial (Objetivo 1).

1.4.1. Responsabilidades Compartidas por Todo el Equipo

- **Comprensión de Requisitos:** Todos los miembros participarán en la discusión y comprensión de los requisitos del cliente.
- **Revisiones de Código:** Se realizarán revisiones cruzadas de código para mejorar la calidad y compartir conocimiento.
- **Pruebas:** Aunque Valentina lidera QA, todos los desarrolladores son responsables de las pruebas unitarias de su código y colaborarán en la identificación y corrección de errores.
- **Reuniones de Equipo:** Participación activa en reuniones periódicas de seguimiento, planificación y resolución de problemas, siguiendo la metodología Scrum.
- **Gestión del Conocimiento:** Compartir aprendizajes y soluciones encontradas durante el desarrollo.
- **Adaptabilidad:** Estar dispuestos a asumir tareas fuera de su rol principal si el proyecto lo requiere, dada la dinámica de un equipo pequeño y un proyecto formativo.

1.5. Plan del Proyecto

En base a la metodología de trabajo expuesta en la Sección 1.3 y las asignaciones de la Sección 1.4, se presenta en las Figuras 1.1 a la 1.4 la Carta Gantt de este proyecto.

Figura 1.1: Carta Gantt del proyecto para el Objetivo Especifico 1

WBS	Nombre	Trabajo
1	**OE1: Formulación y Modelado Conceptual**	**100d**
2	**OE2: Diseño e Implementación de la BD**	**100d**
2.1	**Transformación del Modelo**	**20d**
2.1.1	Tarea A	5d
2.1.2	Tarea B	5d
2.1.3	Tarea C	5d
2.1.4	Tarea D	5d
2.2	**Normalización**	**20d**
2.2.1	Tarea A	5d
2.2.2	Tarea B	2d
2.2.3	Tarea C	3d
2.2.4	Tarea D	5d
2.2.5	Tarea E	5d
2.3	**Scripts SQL**	**20d**
2.3.1	Tarea A	3d
2.3.2	Tarea B	3d
2.3.3	Tarea C	4d
2.3.4	Tarea D	2d
2.3.5	Tarea E	3d
2.3.6	Tarea F	5d
2.4	**Carga de Datos y Validación**	**20d**
2.4.1	Tarea A	10d
2.4.2	Tarea B	5d
2.4.3	Tarea C	2d
2.4.4	Tarea D	3d
2.5	**Consultas y Vistas**	**20d**
2.5.1	Tarea A	2d
2.5.2	Tarea B	3d
2.5.3	Tarea C	5d
2.5.4	Tarea D	5d
2.5.5	Tarea E	5d
3	**OE3: Diseño y Desarrollo del Sistema Web**	**70d**
4	**OE4: Pruebas, Validación y Despliegue**	**71d**

Figura 1.2: Carta Gantt del proyecto para el Objetivo Especifico 2

Figura 1.3: Carta Gantt del proyecto para el Objetivo Especifico 3

Figura 1.4: Carta Gantt del proyecto para el Objetivo Especifico 4

En las Tablas 1.1 a la **??** de presentan las precedencias de cada tarea de esta carta Gantt para cada Objetivo especifico.

WBS	Duración	Después de	Simultanea a	Antes de
A (1.1.1)	5	-	-	B
B (1.1.2)	5	A	H	C,D,E
C (1.1.3)	10	B	D,E	G
D (1.1.4)	5	B	C,E	G
E (1.1.5)	5	B	D,E	F
F (1.1.6)	5	E	-	G
G (1.1.6)	5	C,D,F	-	-
H (1.2.1)	10	A	B	I
I (1.2.2)	10	H	-	J
J (1.2.3)	10	I	-	K
K (1.2.4)	10	J	-	L
L (1.2.5)	15	K	-	M
M (1.2.6)	5	L	-	-

Tabla 1.1: Tabla de Precedencias para el Objetivo Especifico 1

Juaramir: Repetir proceso para completar... **END.**

1.6. Justificación del Proyecto, Utilidad y Aporte para la Universidad

1.6.1. Justificación del Proyecto

La gestión eficiente de la información académica y de investigación es un pilar fundamental para el funcionamiento y la excelencia de cualquier institución de educación superior. En el contexto actual, donde las universidades manejan un volumen creciente y complejo de datos relacionados con su personal docente, alumnado, proyectos de investigación, programas de postgrado y estructura departamental, la necesidad de contar con herramientas tecnológicas robustas y centralizadas se vuelve imperativa.

Actualmente, es común que esta información se encuentre dispersa en múltiples sistemas aislados, hojas de cálculo o incluso en formatos manuales, lo que genera ineficiencias, dificulta la toma de decisiones informadas, incrementa el riesgo de inconsistencias y limita la capacidad de análisis integral del quehacer universitario. El presente proyecto se justifica por la necesidad de superar estas limitaciones mediante el desarrollo de un sistema de base de datos web diseñado a medida, que permita una administración integrada, ágil y precisa de todos estos datos críticos.

1.6.2. Utilidad del Sistema para la Universidad

El sistema de base de datos web propuesto ofrecerá una utilidad transversal a diversas áreas y niveles de la Universidad:

1. **Centralización e Integración de la Información**: Se consolidarán los datos de profesores, alumnos, proyectos, departamentos, programas de postgrado y asignaturas en una única plataforma. Esto eliminará redundancias, mejorará la consistencia de los datos y proporcionará una visión unificada de las actividades académicas y de investigación.
2. **Optimización de Procesos Administrativos**: Se simplificarán y agilizarán numerosas tareas administrativas, tales como:
    - La asignación de profesores a proyectos y la gestión de sus roles (investigador principal, investigador).
    - El seguimiento de la participación de alumnos de postgrado en proyectos como ayudantes de investigación y la asignación de sus supervisores.
    - La administración de la carga docente y la afiliación de profesores a departamentos con sus respectivos porcentajes de dedicación.
    - La gestión de la información de los programas de postgrado y la adscripción de los alumnos. La asignación de directores de departamento y asesores de alumnos.
3. **Facilitación del Acceso a la Información**: Permitirá a los usuarios autorizados (desde personal administrativo hasta directivos y los propios académicos y alumnos, según sus perfiles) acceder de manera rápida y segura a la información que necesitan, mejorando la transparencia y la capacidad de respuesta.
4. **Soporte para la Generación de Informes y Estadísticas**: Al contar con datos estructurados y centralizados, el sistema facilitará la generación de informes detallados y estadísticas relevantes para la evaluación del desempeño, la planificación estratégica y la acreditación institucional. Por ejemplo, se podrá conocer fácilmente la productividad investigadora, la carga académica por departamento o la tasa de participación de alumnos en proyectos.

1.6.3. Aporte del Sistema a la Universidad

La implementación de este sistema representará un aporte significativo para la Universidad en múltiples dimensiones:

1. **Mejora de la Eficiencia Operativa**:
    - Reducción del tiempo dedicado a la búsqueda y consolidación manual de información.
    - Automatización de procesos clave, disminuyendo la carga de trabajo administrativo y la probabilidad de errores humanos.

- Optimización en la asignación de recursos humanos y financieros en proyectos y departamentos.

2. **Fortalecimiento de la Toma de Decisiones Estratégicas**:
    - Proporcionará a la gerencia y a los responsables de unidades académicas información fiable y oportuna para la planificación a corto, mediano y largo plazo.
    - Permitirá identificar fortalezas, debilidades, oportunidades y amenazas en el ámbito académico y de investigación con base en evidencia concreta.
    - Facilitará la evaluación del impacto de programas y políticas institucionales.
3. **Impulso a la Gestión de la Investigación**:
    - Mejorará el seguimiento de los proyectos de investigación: sus participantes, financiamiento, plazos y supervisión.
    - Facilitará la identificación de sinergias y posibles colaboraciones entre investigadores y departamentos.
    - Contribuirá a la visibilidad de la producción científica de la institución.
4. **Mejora en la Gestión y Apoyo Académico**:
    - Optimizará la administración de los programas de postgrado y el seguimiento de sus estudiantes.
    - Facilitará la labor de supervisión y asesoría a los alumnos.
    - Permitirá una gestión más clara de las responsabilidades docentes y de investigación del cuerpo académico.
5. **Incremento de la Transparencia y Rendición de Cuentas**:
    - Proporcionará un registro claro y auditable de las actividades y asignaciones.
    - Facilitará la preparación de informes para organismos de acreditación, entidades financiadoras y la comunidad universitaria en general.
6. **Modernización Tecnológica y Base para el Futuro**:
    - Representa un paso adelante en la modernización de la infraestructura tecnológica de la Universidad.
    - Un sistema bien diseñado y escalable podrá adaptarse a futuras necesidades e integrarse con otras plataformas institucionales, consolidando el ecosistema digital de la Universidad.

En resumen, este sistema no solo resolverá problemas operativos existentes, sino que también se convertirá en una herramienta estratégica que potenciará la capacidad de gestión, la calidad de la investigación y la formación académica, y la eficiencia general de la Universidad, contribuyendo directamente al cumplimiento de su misión y visión institucional.

1.7. Análisis de Viabilidad del Proyecto

La implementación del sistema de base de datos web propuesto para la Universidad se considera viable tras analizar los siguientes aspectos:

1.7.1. Viabilidad Técnica

El proyecto es técnicamente viable por las siguientes razones:

- **Tecnologías Maduras y Disponibles**: Las tecnologías propuestas para el desarrollo (HTML, CSS, JavaScript para el frontend; PHP para el backend; un Sistema Gestor de Base de Datos SQL como MySQL o PostgreSQL) son estándares en la industria, maduras, ampliamente documentadas y con una gran comunidad de soporte. Existen abundantes herramientas de desarrollo, muchas de ellas de código abierto, que facilitan su implementación.
- **Complejidad Manejable**: Si bien el sistema integra diversas entidades y relaciones, la lógica de negocio (operaciones CRUD, gestión de roles, reglas de integridad) es abordable con las tecnologías mencionadas. El modelado relacional y la normalización (hasta 3FN) asegurarán una base de datos robusta y eficiente. La implementación de funciones y triggers (PL/SQL o similar, según el SGBD) está dentro de las capacidades estándar de los SGBD modernos.
- **Recursos Humanos Calificados**: Se asume la disponibilidad de un equipo de desarrollo con experiencia en análisis de sistemas, diseño de bases de datos, desarrollo web full-stack (PHP, HTML, CSS) y gestión de proyectos.
- **Infraestructura Requerida**: La infraestructura necesaria (servidores web y de base de datos) puede ser implementada en las instalaciones de la Universidad o mediante servicios de cloud computing, ofreciendo flexibilidad y escalabilidad. Se considera que la Universidad cuenta con la capacidad para alojar o contratar estos servicios.
- **Escalabilidad y Rendimiento**: El diseño propuesto, incluyendo una correcta normalización, indexación y la posibilidad de optimizar consultas, permitirá que el sistema maneje el volumen de datos y la carga de usuarios esperada en una institución universitaria. El sistema podrá escalar horizontal o verticalmente según las necesidades futuras.

1.7.2. Viabilidad Económica

Se estima que el proyecto es económicamente viable, considerando:

Costos de Desarrollo

- **Personal**: Principal componente de costo, incluyendo analistas, diseñadores, desarrolladores (frontend y backend), especialistas en bases de datos y testers durante los 15 meses.
- **Software**: Potencialmente, costos de licencias para el SGBD o herramientas de desarrollo especializadas, aunque se priorizará el uso de software de código abierto o con licenciamiento favorable para instituciones educativas.
- **Hardware/Infraestructura**: Costos iniciales de adquisición o configuración de servidores, o costos recurrentes si se opta por servicios en la nube.

Costos Operativos (Post-Implementación)

- Mantenimiento del software y la base de datos.
- Soporte técnico a usuarios.
- Costos continuos de infraestructura (energía, conectividad, cloud si aplica).
- Capacitación continua.

Retorno de la Inversión (ROI) y Beneficios

Aunque muchos beneficios son cualitativos, su impacto económico es considerable:

- **Reducción de Costos Directos**: Disminución de horas-hombre dedicadas a tareas manuales de gestión de datos, generación de informes y resolución de inconsistencias.
- **Eficiencia Mejorada**: Optimización de procesos que liberan tiempo de personal cualificado para tareas de mayor valor añadido.
- **Mejora en la Toma de Decisiones**: Decisiones basadas en datos precisos y oportunos pueden llevar a una asignación más eficiente de recursos, optimización de programas académicos y mejora en la captación de fondos de investigación.
- **Cumplimiento y Acreditación**: Un sistema robusto facilita el cumplimiento de normativas y los procesos de acreditación, evitando posibles sanciones o asegurando financiamiento.
- **Reducción de Errores**: La automatización y validaciones disminuyen la incidencia de errores costosos.

La inversión inicial se justifica por los ahorros a mediano y largo plazo, la mejora en la calidad de la gestión y el soporte a las funciones críticas de la Universidad. Se recomienda realizar un análisis costo-beneficio más detallado una vez afinados los requerimientos técnicos específicos y la selección de plataformas.

1.7.3. Viabilidad Operativa

El sistema se considera operativamente viable, ya que:

- **Alineación con Necesidades del Usuario**: El sistema se diseñará basándose en los requisitos específicos proporcionados, buscando resolver problemáticas concretas de la gestión académica y de investigación. Las interfaces diferenciadas para usuarios y administradores con distintos niveles de funcionalidad CRUD están pensadas para adaptarse a sus roles.
- **Aceptación por parte de los Usuarios**: Se mitigarán resistencias al cambio mediante:
  - **Capacitación**: Se planificarán sesiones de capacitación para los diferentes perfiles de usuario.
  - **Diseño Intuitivo**: Se priorizará una interfaz de usuario clara, amigable y fácil de usar.
  - **Participación**: Involucrar a representantes de los usuarios finales durante las fases de diseño y pruebas de aceptación (UAT) para asegurar que el sistema cumpla con sus expectativas y facilite su trabajo diario.
- **Integración en Flujos de Trabajo**: El sistema está concebido para integrarse y optimizar los flujos de trabajo existentes. Es posible que se requiera una adaptación de ciertos procesos manuales para aprovechar al máximo las capacidades del nuevo sistema, lo cual se gestionará como parte del proceso de implementación.
- **Soporte y Mantenimiento**: Se deberá establecer un plan de soporte técnico continuo (interno o externo) para resolver incidencias, realizar mantenimiento preventivo y aplicar futuras actualizaciones o mejoras. La documentación técnica y de usuario generada facilitará estas labores.
- **Seguridad de la Información**: Se implementarán medidas de seguridad adecuadas para proteger la integridad, confidencialidad y disponibilidad de los datos, incluyendo gestión de accesos basada en roles, protección contra amenazas comunes y políticas de respaldo.

1.7.4. Viabilidad de Plazos (Cronograma – 15 Meses)

Un plazo total de 15 meses de trabajo efectivo se considera factible para la realización de este proyecto, dada su envergadura y la adopción de la metodología Scrum. Este marco de trabajo ágil permitirá entregas incrementales y una adaptación continua a lo largo del desarrollo.

La planificación temporal se estructura de la siguiente manera, considerando los períodos de receso académico:

- **Inicio del Proyecto:** Septiembre 2025.
- **Duración Total de Trabajo Efectivo:** 15 meses.

- **Meses No Laborables (Receso Académico):** Febrero 2026 y Agosto 2026.
- **Finalización Estimada del Proyecto:** Enero 2027.

El desarrollo se organizará en Sprints (ciclos de trabajo cortos, por ejemplo, de 4 semanas). Los objetivos específicos se distribuirán en bloques de trabajo de la siguiente forma:

- **Bloque 1: Objetivo Específico 1 (Formulación y Modelado Conceptual).**
  - **Período de Trabajo:** Septiembre 2025 – Enero 2026 (5 meses de trabajo).
  - **Sprints Estimados:** Aproximadamente 5 Sprints.
  - **Enfoque Principal:** Dedicado a la elaboración del informe técnico inicial (incluyendo análisis de viabilidad, justificación, etc.), análisis exhaustivo de requisitos, y el diseño detallado del Modelo Entidad-Relación (MER) y el Modelo Conceptual de la base de datos.
- **Bloque 2: Objetivo Específico 2 (Diseño e Implementación de la Base de Datos).**
  - **Período de Trabajo:** Marzo 2026 – Julio 2026 (5 meses de trabajo, posterior al receso de Febrero 2026).
  - **Sprints Estimados:** Aproximadamente 5 Sprints.
  - **Enfoque Principal:** Centrado en la transformación del modelo conceptual al modelo relacional, aplicación de técnicas de normalización (hasta 3FN), escritura de los scripts DDL para la creación de tablas y restricciones, carga de datos de prueba, y la implementación de índices y vistas predefinidas para optimizar consultas.
- **Bloque 3: Objetivos Específicos 3 y 4 (Diseño y Desarrollo del Sistema Web, Integración con la Base de Datos, Procesos CRUD, Pruebas y Validación).**
  - **Período de Trabajo:** Septiembre 2026 – Enero 2027 (5 meses de trabajo, posterior al receso de Agosto 2026).
  - **Sprints Estimados:** Aproximadamente 5 Sprints.
  - **Enfoque Principal:** Esta fase comprenderá el diseño de maquetas del sistema web, el desarrollo frontend (HTML, CSS) y backend (PHP), la conexión con la base de datos implementada, y la programación de las funcionalidades CRUD. De forma paralela e integrada en los Sprints, se llevará a cabo el Objetivo 4, realizando pruebas unitarias, de integración, de sistema, y las pruebas de aceptación del usuario (UAT). Se finalizará con la corrección de errores, la elaboración de la documentación final y la preparación para el despliegue.

Consideraciones para cumplir el plazo: El cumplimiento de este cronograma dependerá de varios factores críticos:

- **Gestión de Riesgos Eficaz:** Identificación temprana y mitigación proactiva de posibles riesgos, como cambios no controlados en el alcance (scope creep), retrasos en la toma de decisiones o validaciones por parte del "cliente" (en este caso, el profesorado del curso), o la aparición de problemas técnicos imprevistos.
- **Dedicación y Compromiso del Equipo:** Es fundamental contar con la dedicación adecuada del equipo de desarrollo estudiantil y un compromiso con los objetivos de cada Sprint y las entregas planificadas.
- **Aplicación Rigurosa de Scrum:** La correcta implementación de las ceremonias (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective) y artefactos (Product Backlog, Sprint Backlog, Incremento) de Scrum facilitará la transparencia, la inspección y la adaptación continua, elementos clave para el manejo eficiente del tiempo.
- **Comunicación Constante y Fluida:** Mantener una comunicación efectiva dentro del equipo de desarrollo y con el "cliente" para asegurar la alineación y la rápida resolución de dudas o problemas que puedan surgir.

En conclusión, el plazo de 15 meses de trabajo efectivo, distribuido según el calendario propuesto y gestionado mediante la metodología ágil Scrum, ofrece un marco temporal realista y estructurado para la finalización exitosa del proyecto por parte del equipo de estudiantes.

1.8. Metodología de Desarrollo Propuesta: Scrum

Para un equipo de 6 estudiantes de Ingeniería Civil en Informática que están comenzando a conocer las metodologías de desarrollo de software y que tienen un proyecto de 15 meses, la metodología más apropiada sería **Scrum**.

Justificación de la Elección de Scrum

Scrum es un marco de trabajo ágil que se adapta muy bien a las características de este proyecto y del equipo por las siguientes razones:

1. **Desarrollo Iterativo e Incremental:** Scrum divide el proyecto en ciclos cortos llamados *Sprints* (generalmente de 2 a 4 semanas). Al final de cada Sprint, el equipo entrega un incremento funcional del producto. Esto es ideal para un proyecto de 15 meses, ya que permite al equipo mostrar progreso tangible de forma regular, obtener retroalimentación temprana del "cliente" (profesor del curso) y ajustar el rumbo si es necesario. Para estudiantes, ver resultados funcionales de forma periódica es altamente motivador.
2. **Adaptabilidad y Flexibilidad:** Aunque los requisitos iniciales están definidos, es probable que surjan nuevos entendimientos o se necesiten ajustes, especialmente en la interfaz de usuario y la experiencia de usuario (UI/UX). Scrum permite la adaptación a cambios entre Sprints, incorporando nuevos requisitos o modificaciones en el *Product Backlog*.

3. **Estructura y Claridad para Equipos Nuevos:** Para estudiantes que recién se inician en metodologías, Scrum ofrece un marco con roles, eventos y artefactos definidos que proporcionan una estructura clara:
   - *Roles Adaptados:* Para el contexto estudiantil, estos roles pueden ser:
     - **Product Owner (Dueño del Producto):** Rol asumido idealmente por el "cliente" (profesor) o, de forma interna y en representación del cliente, por un miembro del equipo (ej. el Jefe de Proyecto, Javier Rojas) encargado de definir y priorizar los ítems del Product Backlog.
     - **Scrum Master (Facilitador):** Rol que podría ser asumido por el Jefe de Proyecto (Javier Rojas), enfocándose en asegurar que el equipo siga las prácticas de Scrum, facilitar las reuniones y eliminar impedimentos.
     - **Development Team (Equipo de Desarrollo):** Los 6 estudiantes, como un equipo auto-organizado y multidisciplinario, responsable de entregar el incremento del producto.
   - *Eventos (Reuniones):*
     - *Sprint Planning:* Planificación del trabajo a realizar en el Sprint.
     - *Daily Scrum:* Reunión diaria corta para sincronizar al equipo.
     - *Sprint Review:* Demostración del incremento del producto y recolección de feedback.
     - *Sprint Retrospective:* Reflexión sobre el Sprint anterior para identificar mejoras en el proceso.

     Estas reuniones estructuradas ayudan a mantener el enfoque y la comunicación.
   - *Artefactos:*
     - *Product Backlog:* Lista priorizada de todas las funcionalidades y requisitos del proyecto.
     - *Sprint Backlog:* Conjunto de ítems del Product Backlog seleccionados para un Sprint, más el plan para entregarlos.
     - *Incremento del Producto:* La suma de todos los ítems del Product Backlog completados durante un Sprint y Sprints anteriores, que debe ser potencialmente entregable.

   Esta estructura es fundamental para guiar al equipo a lo largo de los 15 meses de proyecto.

4. **Fomenta la Colaboración y Comunicación:** El tamaño del equipo (6 personas) es ideal para Scrum. Las reuniones como el Daily Scrum y la planificación y revisión conjuntas de los Sprints aseguran que todos los miembros estén alineados, compartan información y colaboren estrechamente para alcanzar el objetivo del Sprint.

5. **Gestión de Riesgos Temprana:** La naturaleza iterativa y las revisiones constantes (Sprint Review, Sprint Retrospective) permiten identificar problemas, riesgos o malentendidos de

forma temprana, en lugar de esperar hasta el final del proyecto. Esto es crucial en un proyecto de 15 meses, donde los problemas no detectados pueden escalar significativamente.

6. **Enfoque en la Entrega de Valor:** Cada Sprint se enfoca en entregar las funcionalidades de mayor valor para el cliente (priorizadas en el Product Backlog), lo que asegura que el sistema evolucione de manera útil y relevante desde las primeras etapas.

7. **Oportunidad de Aprendizaje Práctico:** Implementar Scrum proporcionará al equipo una valiosa experiencia práctica con una de las metodologías ágiles más utilizadas en la industria del software. Esto constituye un objetivo de aprendizaje importante en su formación como Ingenieros Civiles en Informática, preparándolos para entornos de trabajo reales.

Aunque otras metodologías ágiles como Kanban podrían ser consideradas por su simplicidad visual y enfoque en el flujo, la estructura de Sprints de Scrum, junto con sus roles y ceremonias definidas, ofrece un andamiaje más robusto y guiado para un equipo de estudiantes que se enfrenta a un proyecto de duración considerable y que necesita aprender a gestionar su trabajo de forma sistemática, colaborativa y adaptativa. Les ayudará a mantener un ritmo sostenible y a asegurar entregas consistentes y de valor a lo largo de los 15 meses del proyecto.

1.9. Planificación Temporal del Desarrollo del Proyecto (Metodología Scrum)

Consideraciones Generales de la Planificación:

- **Inicio del Proyecto:** Septiembre 2025.
- **Duración Total:** 15 meses de trabajo efectivo.
- **Meses No Laborables (Receso Académico):** Febrero 2026 y Agosto 2026 no se consideran meses de trabajo.
- **Sprints:** Se asumen Sprints de 4 semanas de duración. Un mes de trabajo tiene aproximadamente un Sprint.
- **Planificación Adaptativa:** El contenido exacto de cada Sprint (Sprint Backlog) será definido por el equipo al inicio de cada uno, tomando tareas del Product Backlog priorizado. La Tabla 1.2 ofrece una guía de alto nivel sobre el foco de cada Sprint.

Notas Importantes para el Equipo:

- **Flexibilidad del Product Backlog:** Aunque se presenta un foco, el equipo priorizará las tareas del Product Backlog al inicio de cada Sprint (Sprint Planning) según el valor que aporten y el estado del proyecto.

- **Reuniones Scrum:** Se deben llevar a cabo todas las ceremonias de Scrum: Sprint Planning, Daily Scrums, Sprint Review y Sprint Retrospective.
- **Testing Continuo:** Aunque el Bloque 3 tiene un fuerte componente del Objetivo 4 (Pruebas), las pruebas unitarias y de integración deben ser una actividad continua desde que se empieza a escribir código.
- **Objetivo del Sprint:** Cada Sprint debe tener un "Sprint Goal" claro, que es un objetivo conciso de lo que el Sprint intentará lograr.

Tabla 1.2: Planificación Temporal del Proyecto (Metodología Scrum)

Fase / Bloque de Objetivos	Mes de Trabajo	Mes y Año Calendario	Sprint #	Duración Estimada del Sprint	Foco Principal y Actividades Clave del Sprint (basado en Objetivos)	Entregables Clave del Sprint (Ejemplos)
Bloque 1: Obj. Específico 1 (Formulación y Modelado Conceptual)	1	Sep 2025	1	4 semanas	Inicio del proyecto, configuración del equipo y entorno Scrum. Revisión detallada de requisitos. Estructura del informe técnico. Planificación inicial del Product Backlog.	Acta de constitución del proyecto (simplificada), Plan de trabajo inicial, Product Backlog inicial.
Bloque 1: Obj. Específico 1 (Formulación y Modelado Conceptual)	2	Oct 2025	2	4 semanas	Redacción secciones clave del informe (Introducción, Objetivos, Justificación, Viabilidad, Metodología). Identificación inicial de entidades y atributos.	Borrador avanzado del informe técnico. Lista de entidades y atributos preliminares.
Bloque 1: Obj. Específico 1 (Formulación y Modelado Conceptual)	3	Nov 2025	3	4 semanas	Detalle de atributos. Identificación de relaciones y cardinalidad. Inicio del diseño del Diagrama Entidad-Relación (DER).	Especificación detallada de atributos. Borrador inicial del DER.
Bloque 1: Obj. Específico 1 (Formulación y Modelado Conceptual)	4	Dic 2025	4	4 semanas	Finalización y refinamiento del DER. Documentación del Modelo Entidad-Relación (MER).	Diagrama Entidad-Relación (DER) finalizado. Documentación completa del MER.
Bloque 1: Obj. Específico 1 (Formulación y Modelado Conceptual)	5	Ene 2026	5	4 semanas	Validación final del MER y del informe técnico con el "cliente" (profesor). Consolidación del informe. Preparación para el Bloque 2. Refinamiento del Product Backlog para Obj. 2.	Informe técnico final (Obj. 1). Modelo Conceptual de Datos aprobado.

Continuará en la página siguiente...

Tabla 1.2: Planificación Temporal del Proyecto (Metodología Scrum) (Continuación)

Fase / Bloque de Objetivos	**Mes de Trabajo**	**Mes y Año Calendario**	**Sprint #**	**Duración Estimada del Sprint**	**Foco Principal y Actividades Clave del Sprint (basado en Objetivos)**	**Entregables Clave del Sprint (Ejemplos)**
Receso Académico	–	*Feb 2026*	–	–	*Mes no laborable*	–
Bloque 2: Obj. Específico 2 (Diseño e Implementación de BD)	6	Mar 2026	6	4 semanas	Transformación del MER al Modelo Relacional. Inicio de Normalización (1FN, 2FN).	Modelo Relacional preliminar. Tablas identificadas con atributos y PK/FK iniciales.
Bloque 2: Obj. Específico 2 (Diseño e Implementación de BD)	7	Abr 2026	7	4 semanas	Finalización de Normalización (3FN). Diseño detallado de tablas, tipos de datos y restricciones de integridad.	Modelo Relacional Normalizado (3FN) y documentado. Especificaciones detalladas para scripts DDL.
Bloque 2: Obj. Específico 2 (Diseño e Implementación de BD)	8	May 2026	8	4 semanas	Desarrollo de scripts DDL (`CREATE TABLE`). Creación de la estructura de la base de datos en el SGBD elegido.	Scripts DDL funcionales. Base de datos creada (estructura vacía).
Bloque 2: Obj. Específico 2 (Diseño e Implementación de BD)	9	Jun 2026	9	4 semanas	Diseño y carga de un conjunto inicial de datos de prueba representativos. Verificación de consistencia de datos y restricciones.	Base de datos poblada con datos de prueba. Informe de consistencia de datos.

Continuará en la página siguiente...

Tabla 1.2: Planificación Temporal del Proyecto (Metodología Scrum) (Continuación)

Fase / Bloque de Objetivos	**Mes de Trabajo**	**Mes y Año Calendario**	**Sprint #**	**Duración Estimada del Sprint**	**Foco Principal y Actividades Clave del Sprint (basado en Objetivos)**	**Entregables Clave del Sprint (Ejemplos)**
Bloque 2: Obj. Específico 2 (Diseño e Implementación de BD)	10	Jul 2026	10	4 semanas	Implementación de Índices en columnas clave. Diseño y creación de Consultas y Vistas predefinidas importantes. Evaluación inicial de rendimiento. Preparación para el Bloque 3.	Índices creados. Vistas y consultas predefinidas funcionales.
Receso Académico	–	*Ago 2026*	–	–	*Mes no laborable*	–
Bloque 3: Obj. Específicos 3 y 4 (Desarrollo Web, Pruebas)	11	Sep 2026	11	4 semanas	**O3:** Configuración del entorno de desarrollo web. Diseño de maquetas (wireframes/mockups) V1. Inicio desarrollo Frontend (HTML/CSS estructura base). **O4:** Elaboración del Plan de Pruebas detallado.	Maquetas V1. Estructura base del frontend. Plan de Pruebas V1.
Bloque 3: Obj. Específicos 3 y 4 (Desarrollo Web, Pruebas)	12	Oct 2026	12	4 semanas	**O3:** Desarrollo Frontend (vistas principales). Inicio desarrollo Backend (PHP, conexión BD). Implementación CRUD básico para 1-2 entidades. **O4:** Diseño de casos de prueba (unitarios, integración). Ejecución de pruebas unitarias del código desarrollado.	Vistas principales del frontend. CRUD funcional para 1-2 entidades. Casos de prueba documentados.

Continuará en la página siguiente...

Tabla 1.2: Planificación Temporal del Proyecto (Metodología Scrum) (Continuación)

Fase / Bloque de Objetivos	Mes de Trabajo	Mes y Año Calendario	Sprint #	Duración Estimada del Sprint	Foco Principal y Actividades Clave del Sprint (basado en Objetivos)	Entregables Clave del Sprint (Ejemplos)
Bloque 3: Obj. Específicos 3 y 4 (Desarrollo Web, Pruebas)	13	Nov 2026	13	4 semanas	**O3:** Desarrollo Frontend (interfaces Admin/Usuario). Desarrollo Backend (más entidades CRUD, lógica de negocio). Implementación de Funciones/Triggers PL/SQL (o similar) según necesidad. **O4:** Ejecución de Pruebas de Integración (Frontend-Backend-DB).	Interfaces Admin/Usuario funcionales para módulos desarrollados. Funciones/Triggers implementados. Informe de pruebas de integración.
Bloque 3: Obj. Específicos 3 y 4 (Desarrollo Web, Pruebas)	14	Dic 2026	14	4 semanas	**O3:** Finalización desarrollo de funcionalidades principales. Refinamiento UI/UX. Integración completa. **O4:** Ejecución de Pruebas de Sistema. Inicio Pruebas de Aceptación del Usuario (UAT) con “cliente”. Corrección de errores identificados.	Sistema integrado con funcionalidades clave. Informe de pruebas de sistema. Feedback inicial de UAT.
Bloque 3: Obj. Específicos 3 y 4 (Desarrollo Web, Pruebas)	15	Ene 2027	15	4 semanas	**O3:** Últimos ajustes basados en feedback de UAT. **O4:** Finalización de UAT. Corrección de errores finales. Preparación para el despliegue (documentación final: manual de usuario, manual técnico). Sprint Review y Retrospectiva final del proyecto.	Sistema validado y listo para “entrega”. Documentación final completa. Lecciones aprendidas del proyecto.

1.10. Diagramas UML

Capítulo 2

Modelo de Datos

2.1. Requisitos

A continuación se establecen los requisitos del sistema de datos solicitado por la universidad.

1. Cada profesor tiene RUN, nombre, edad, rango y especialidad de investigación.
2. Cada proyecto tiene número de proyecto, nombre de patrocinador (por ejemplo, el CSIC), fecha de comienzo, fecha de finalización y presupuesto.
3. Cada alumno tiene RUN, nombre, edad.
4. Cada alumno puede o no pertenecer a un programa de postgrado (por ejemplo, magíster o doctorado), del cual interesa el nombre y código.
5. Cada proyecto está dirigido por un profesor (conocido como investigador principal de ese proyecto).
6. En cada proyecto trabajan uno o varios profesores (conocidos como investigadores de ese proyecto).
7. Cada profesor puede dirigir varios proyectos o trabajar en ellos.
8. En cada proyecto trabajan uno o varios alumnos de postgrado (conocidos como ayudantes de investigación de ese proyecto).
9. Cuando los alumnos de postgrado trabajan en un proyecto, su trabajo debe supervisarlo un profesor. Cada alumno de postgrado puede trabajar en varios proyectos, en cuyo caso puede tener un supervisor diferente en cada uno de ellos.
10. Cada departamento tiene número de departamento, nombre de departamento y despacho principal.

11. Cada departamento tiene un profesor (conocido como director) que lo dirige.
12. Cada profesor trabaja en uno o varios departamentos, y por cada departamento en que trabaja se asocia un porcentaje de tiempo a su trabajo.
13. Cada alumno de postgrado tiene un departamento principal en el que trabaja en su titulación.
14. Cada alumno de postgrado tiene otro alumno de postgrado más veterano (conocido como asesor del alumno) que lo aconseja sobre las asignaturas en las que debe matricularse.

2.2. Modelo Entidad Relación

2.2.1. Definición de Entidades

Herencia

A partir de los requisitos expuestos en las líneas 1 y 2, se identifican las entidades `Profesor` y `Alumno`. Ambas comparten atributos inherentes al concepto de `Persona`, por lo que se propone utilizar una jerarquía de herencia para optimizar el modelo de datos, como se ilustra en la Figura 2.1. Dado que los únicos roles definidos en el sistema son `Profesor` y `Alumno`, y no se contempla la posibilidad de que un individuo ocupe ambos roles simultáneamente, se define esta especialización como **total y sin solapamiento**. Esto implica que cada instancia de `Persona` registrada en el sistema será exclusivamente un `Profesor` o un `Alumno`.

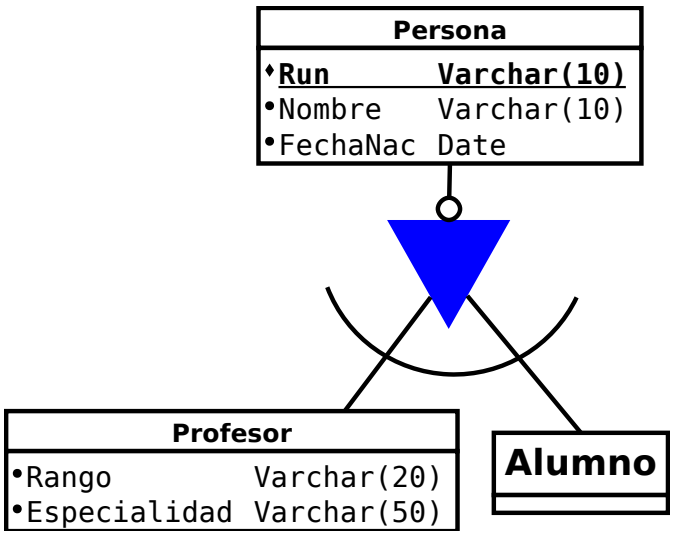

Figura 2.1: Herencia Persona

Entidades individuales

A partir del requisito de la línea 2, se define la entidad `Proyecto`. La estructura detallada de esta entidad, incluyendo sus atributos y características, se presenta en la Tabla 2.1.

Proyecto	
Numero (PK)	Integer
Patrocinador	Varchar (20)
FechaInicio	Date
FechaFin	Date
Presupuesto	Money

Tabla 2.1: Entidad Proyecto

El requerimiento expuesto en la línea 4 da origen a la entidad `Postgrado`, cuya composición y atributos se desglosan en la Tabla 2.2.

Postgrado	
Codigo (PK)	Integer
Nombre	Varchar (30)

Tabla 2.2: Entidad Postgrado

La entidad `Departamento`, identificada a partir del requerimiento de la línea 10, se detalla en la Tabla 2.3, donde se especifican sus atributos y características.

Departamento	
Numero (PK)	Integer
Nombre	Varchar (30)
Despacho	Integer

Tabla 2.3: Entidad Departamento

2.2.2. Definición de Relaciones

Relaciones n-aria

A partir de los requisitos de las líneas 6 y 9, se define la relación ternaria `Supervisar` que vincula a `Profesor`, `Alumno` y `Proyecto` (ver Figura 2.2). Aunque un profesor puede supervisar a varios alumnos y un proyecto puede involucrar a muchos, el requisito 9 impone una restricción clave: la supervisión de un alumno en un proyecto específico es realizada por un único profesor. Esto refina la cardinalidad general, estableciendo una dependencia funcional donde el par (`Alumno`, `Proyecto`) determina al `Profesor` supervisor.

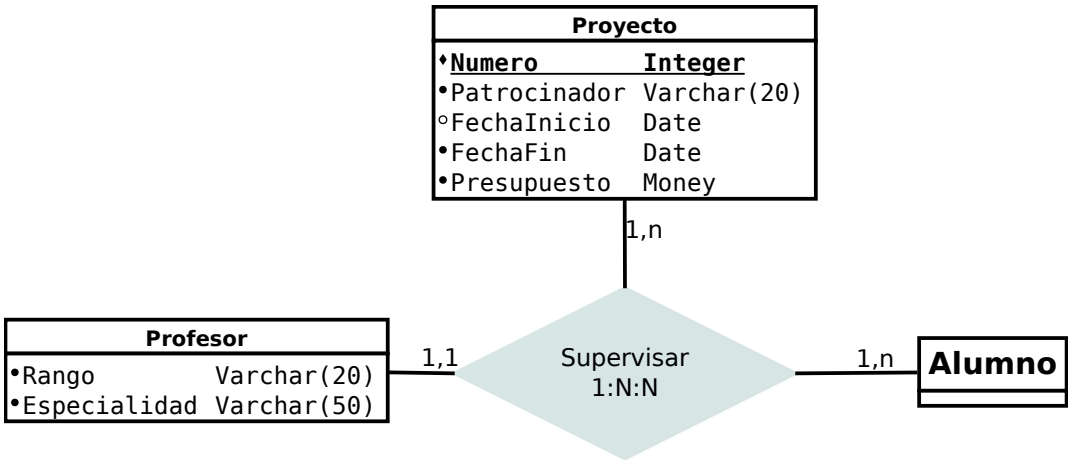

Figura 2.2: Relación Supervisar

Relaciones Muchos a Muchos (N:N)

Si bien el requisito de la línea 5 sugiere que un proyecto es dirigido por 'un' profesor, para dotar al sistema de mayor flexibilidad y capacidad a futuro, se opta por modelar la relación `Dirigir` como muchos a muchos (N:M). Esta decisión de diseño, ilustrada en la Figura 2.3, permite que el sistema registre múltiples directores o co-directores por proyecto si fuera necesario, cumpliendo a la vez con el requisito de la línea 7 que establece que un profesor puede dirigir varios proyectos. Este enfoque proporciona mayor adaptabilidad a casos de uso complejos.

Figura 2.3: Relación Dirigir

El requisito de la línea 12 establece que un profesor puede trabajar en varios departamentos y que a esta vinculación se le asocia un "porcentaje de tiempo". Para modelar esto correctamente, se define la relación `Laborar` con una cardinalidad de muchos a muchos (N:M) entre `Profesor` y `Departamento`. Esta relación, ilustrada en la Figura 2.4, incluye el atributo `porcentaje` que depende de la combinación de un profesor y un departamento específicos.

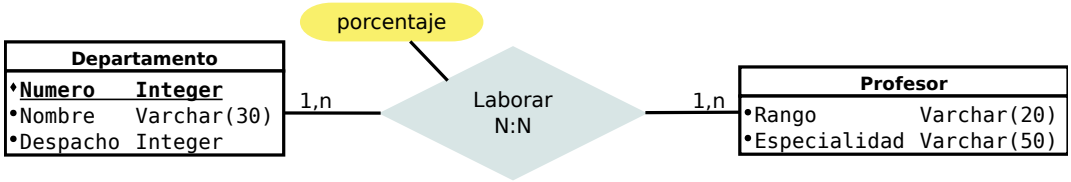

Figura 2.4: Relación Laborar

A partir del requisito expuesto en la línea 8, se establece la relación `Trabajar` entre las entidades `Proyecto` y `Alumno`. Se trata de una relación con cardinalidad muchos a muchos (N:M), ya que un proyecto puede tener "uno o varios" alumnos trabajando en él, y (como se complementa en la línea 9) un alumno puede trabajar en varios proyectos (Ver Figura 2.5).

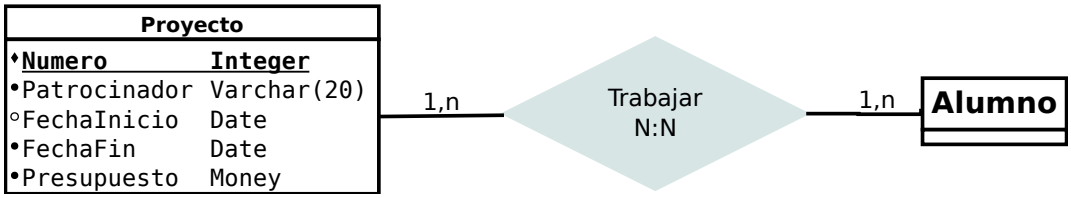

Figura 2.5: Relación Trabajar

Relaciones Uno a Muchos (1:N)

La relación `Cursar`, que vincula a un `Alumno` con un `Postgrado`, se deriva del requisito de la línea 4. Este define una cardinalidad de uno a muchos (1:N), donde un `Postgrado` puede tener múltiples alumnos inscritos, pero un `Alumno` solo puede pertenecer a un programa de postgrado a la vez, siendo esta participación opcional. La Figura 2.6 ilustra esta relación y sus respectivas cardinalidades (0,1) para `Alumno` y (0,N) para `Postgrado`.

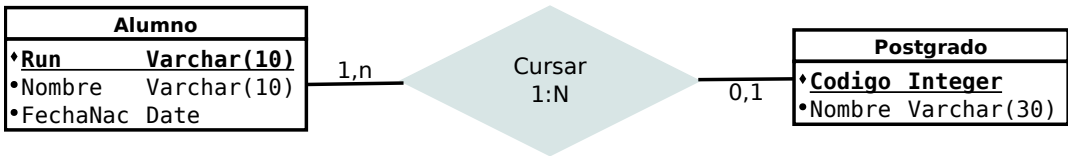

Figura 2.6: Relación Cursar

A partir del requisito de la línea 13, se define la relación `Pertenece` como una asociación de uno a muchos (1:N) que vincula a `Departamento` con `Alumno`. Como se ilustra en la Figura **??**, esta cardinalidad refleja que, si bien un departamento puede tener múltiples alumnos (1,N), cada alumno de postgrado debe estar asociado obligatoriamente a un único departamento principal (1,1).

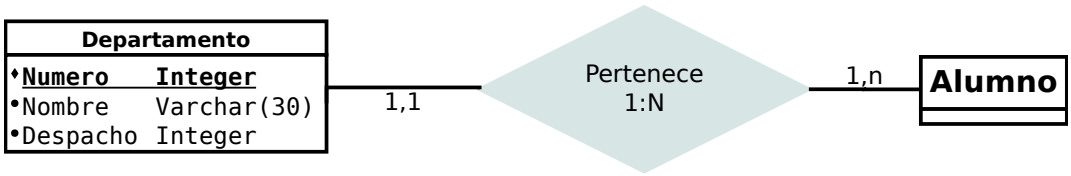

Figura 2.7: Relación Pertenece

Relaciones Uno a Uno (1:1)

A partir del requisito de la línea 11, se define la relación `Administrar` como una asociación de uno a uno (1:1) que vincula a un `Departamento` con su `Profesor` director. La Figura 2.8 ilustra esta estructura, donde la participación del `Departamento` es obligatoria y única (1,1), mientras que la del `Profesor` en este rol es opcional y, como máximo, única (0,1).

Figura 2.8: Relación Administrar

Relaciones Reflexivas

El requisito de la línea 14, que establece que cada alumno tiene un asesor más veterano, da origen a la relación reflexiva `Asesorar` en la entidad `Alumno`. Se trata de una relación de uno a muchos (1:N), donde un alumno en el rol de .ªsesor"puede guiar a múltiples alumnos .ªsesorados", mientras que cada asesorado está vinculado obligatoriamente a un único asesor. Esta estructura recursiva se representa en la Figura 2.9.

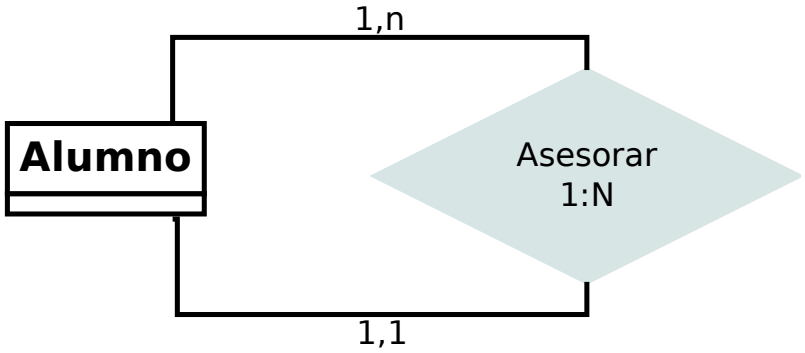

Figura 2.9: Relación Asesorar

2.2.3. Modelo Entidad Resultante

Una vez establecidas las entidades y sus atributos como se presentó en la sección 2.2.1 y las relaciones y herencia presentadas en la sección 2.2.2, se obtiene el modelo Entidad Relación completo que se ve en la Figura 2.10.

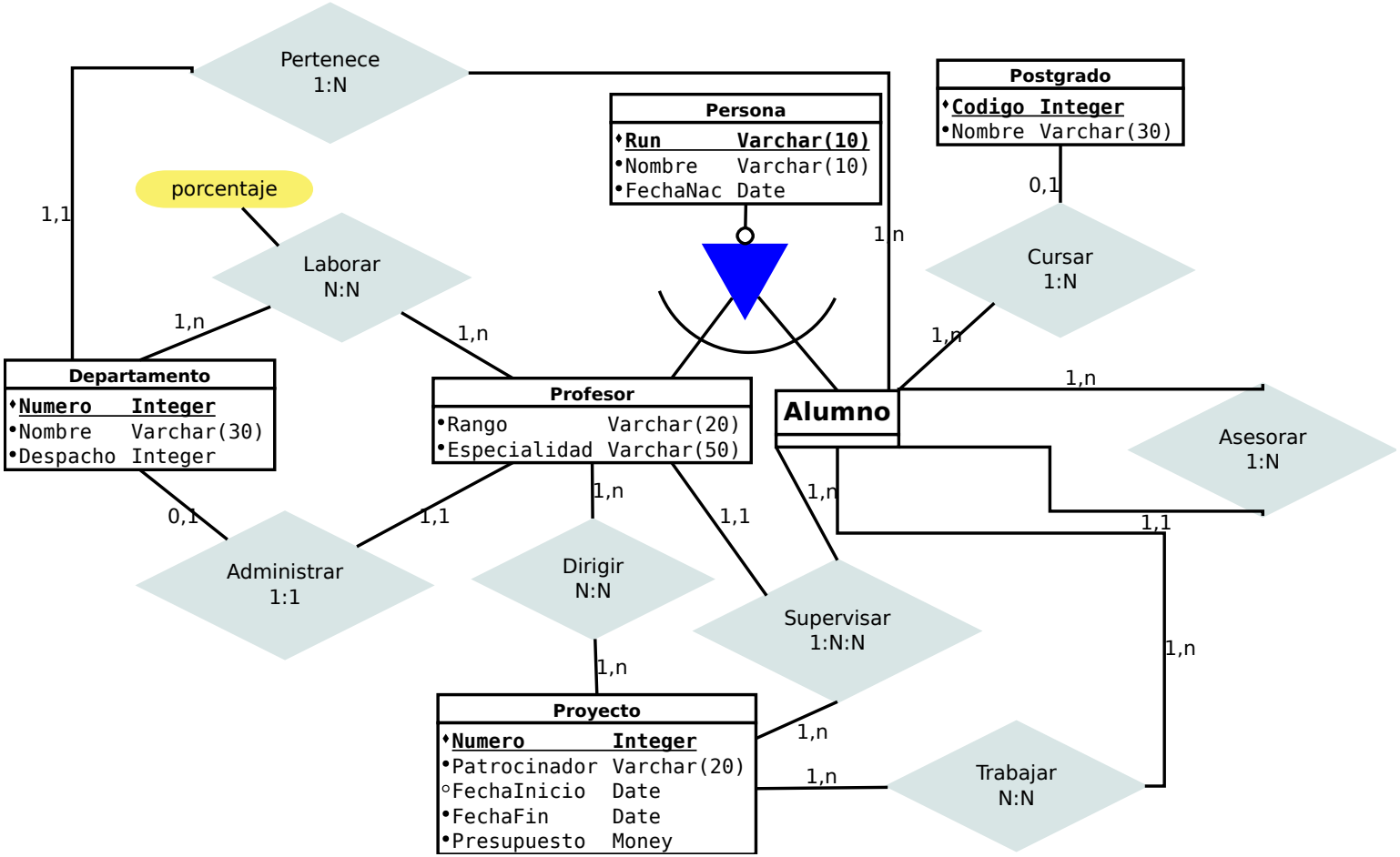

Figura 2.10: Modelo Entidad Relación de la universidad

2.2.4. Modelo Entidad Relación sin Herencia

Considerando el Modelo Entidad Relación de la Figura 2.10 se aprecia que se implementó una herencia total sin solapamiento, lo que implica que para eliminarla se debe hacer eliminación del supertipo, por lo que los atributos del padre como sus relaciones pasan ahora a las entidades hijas, con esto obtenemos el modelo entidad relación final que se ve en la Figura 2.11.

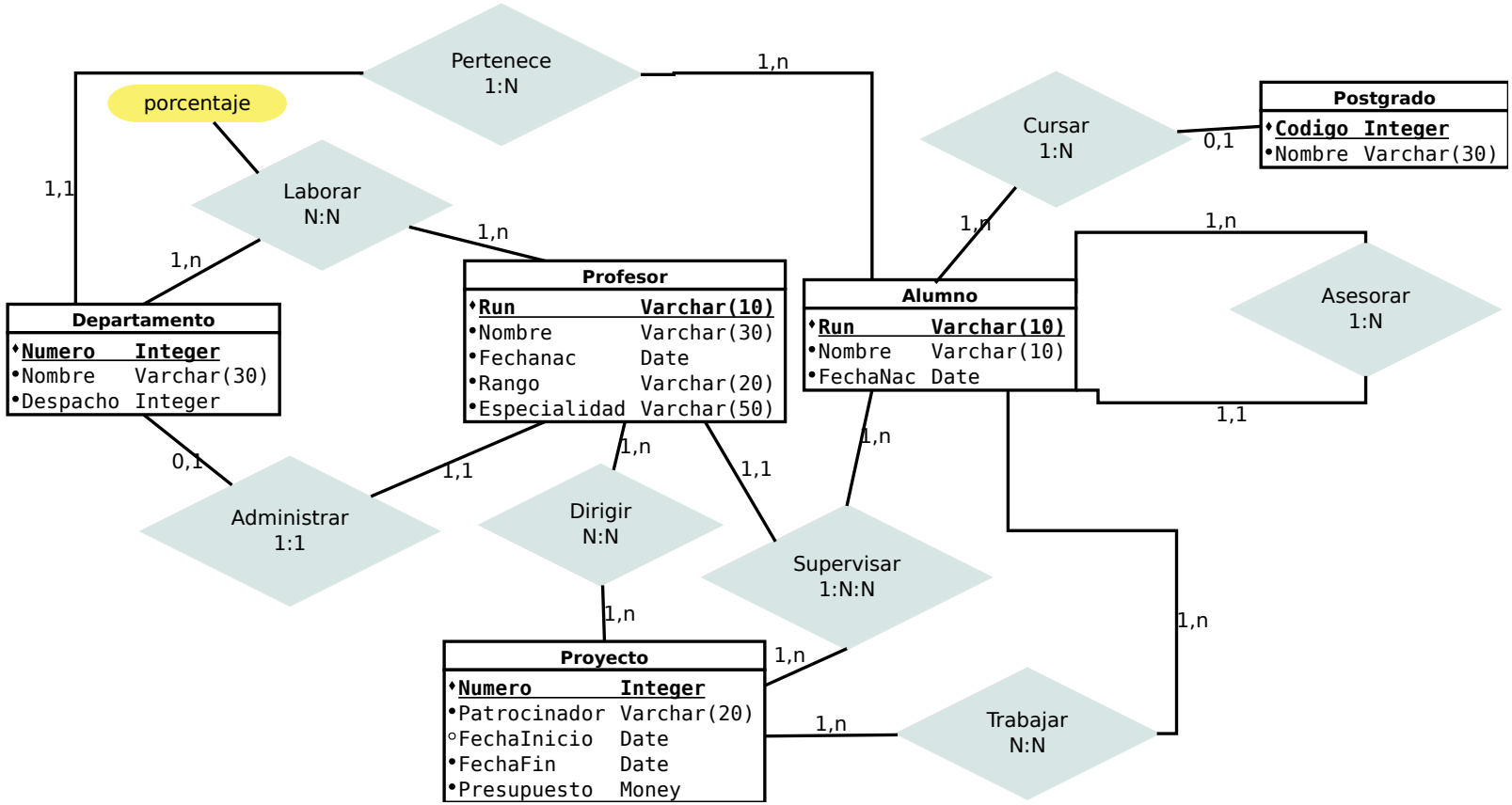

Figura 2.11: MER de la universidad aplicado Eliminación del supertipo a la herencia

2.3. Modelo Relacional

En esta etapa se procede con la conversión del Modelo Entidad-Relación (MER) al Modelo Relacional (MR). El MER de referencia, en el que se ha resuelto la especialización/generalización, se muestra en la Figura 2.11. La siguiente descripción se enfocará exclusivamente en las tablas generadas a partir de relaciones complejas (N:M, ternarias) o aquellas que requirieron modificaciones estructurales. Las entidades que no se detallen a continuación se mapean a tablas de forma directa, conservando su estructura original.

2.3.1. Conversión Relaciones N-arias

Supervisar

Al transformar el Modelo Entidad-Relación (MER) al Modelo Relacional (MR), la relación ternaria `Supervisar` (Figura 2.12) se convierte en una nueva tabla. Esta tabla, también llamada `Supervisar`, incluye las claves foráneas de las tres entidades participantes: `Proyecto`, `Alumno` y `Profesor`.

La clave primaria para esta nueva tabla se define a partir de la dependencia funcional del requisito 9: como un par (`Alumno`, `Proyecto`) determina a un único `Profesor` supervisor, la clave primaria de la tabla `Supervisar` se compone únicamente de las claves foráneas de `Alumno` y `Proyecto`. La clave del `Profesor`, al ser determinada por las otras dos, se mantiene como un atributo no clave. La estructura resultante se detalla en la Tabla 2.4.

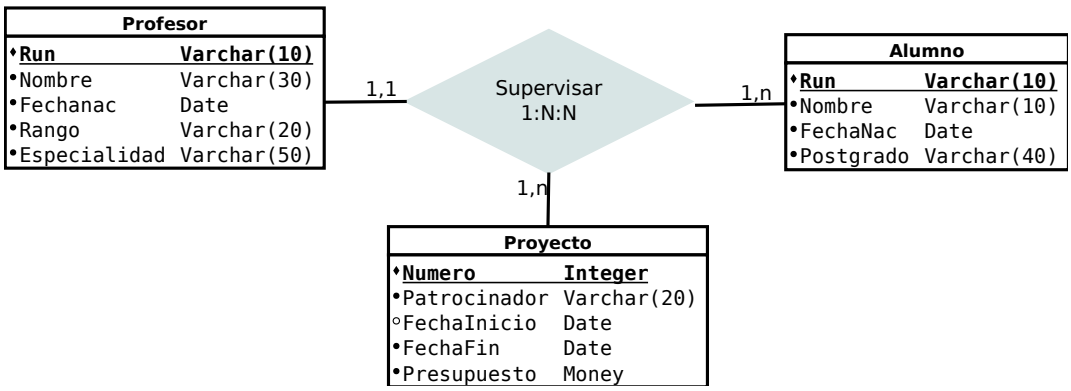

Figura 2.12: Relación Supervisar

Supervisar	
Proyecto_id (PFK)	Integer
Alumno_id (PFK)	Varchar(10)
Profesor_id (FK)	Varchar(10)

Tabla 2.4: MR de la Relación Supervisar

2.3.2. Conversión Relaciones N:N

Dirigir

En el proceso de conversión del Modelo Entidad-Relación (MER) al Modelo Relacional (MR), la relación de muchos a muchos (N:M) `Dirigir`, ilustrada en la Figura 2.13, se transforma en una nueva tabla. Esta tabla, también denominada `Dirigir`, hereda como claves foráneas las claves primarias de las entidades que relaciona (`Proyecto` y `Profesor`), y la combinación de ambas conforma su nueva clave primaria. La estructura final se presenta en la Tabla 2.5.

Figura 2.13: Relación Dirigir

Dirigir	
Proyecto_id (PFK)	Integer
Profesor_id (PFK)	Varchar(10)

Tabla 2.5: MR de la Relación Dirigir

Laborar

En el proceso de conversión del MER al Modelo Relacional, la relación N:M `Laborar` (Figura 2.14) da origen a una nueva tabla. Dicha tabla, llamada `Laborar`, hereda las claves primarias de `Profesor` y `Departamento` para formar su propia clave primaria compuesta. Adicionalmente, se incluye como atributo no clave el `Porcentaje` que pertenecía a la relación. La estructura resultante se detalla en la Tabla 2.6.

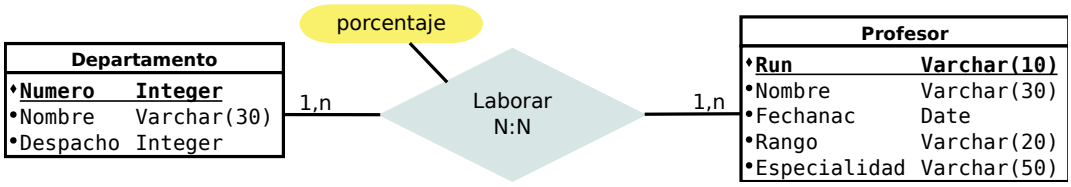

Figura 2.14: Relación Laborar

Laborar	
Departameno_id (PFK)	Integer
Profesor_id (PFK)	Varchar(10)
Porcentaje	Integer

Tabla 2.6: MR de la Relación Laborar

Trabajar

La relación `Trabajar`, representada en la Figura 2.15, es una relación de cardinalidad muchos a muchos (N:M) entre `Proyecto` y `Alumno`. Durante el proceso de conversión del Modelo Entidad-Relación (MER) al Modelo Relacional (MR), este tipo de relación se transforma en una nueva tabla. Para este caso, dicha transformación da origen a la tabla `Trabajar`, presentada en la Tabla 2.7.

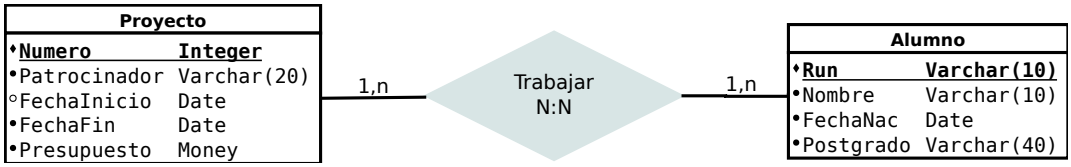

Figura 2.15: Relación Trabajar

Trabajar	
Proyecto_id (PFK)	Integer
Alumno_id (PFK)	Varchar(10)

Tabla 2.7: MR de la Relación Trabajar

2.3.3. Conversión Relaciones 1:N

Cursar

Al transformar el Modelo Entidad-Relación (MER) al Modelo Relacional (MR), la relación de uno a muchos (1:N) `Cursar` (Figura 2.16) se resuelve siguiendo la regla estándar para este tipo de cardinalidad, la cual no requiere la creación de una tabla nueva para la relación.

El procedimiento correcto consiste en propagar la clave primaria de la entidad del lado “1” de la relación (`Postgrado`) a la tabla de la entidad del lado “N” (`Alumno`). Específicamente:

- Se añade una columna de clave foránea en la tabla `Alumno` que haga referencia a la clave primaria de `Postgrado`.

- Dado que el requisito de la línea 4 indica que la participación del alumno es opcional ('puede o no pertenecer'), esta nueva columna de clave foránea en la tabla `Alumno` debe permitir valores `NULL`.

Este enfoque es más eficiente y evita la creación de una tabla adicional innecesaria. La Tabla 2.8 muestra la estructura de la tabla `Alumno` modificada para incluir esta clave foránea.

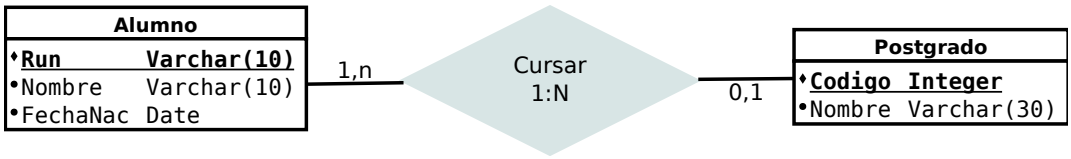

Figura 2.16: Relación Cursar

Alumno	
Run (PK)	Varchar(10)
Nombre	Varchar(40)
Fechanac	Date
Postgrado_id (FK)	Integer

Tabla 2.8: MR de la Relación Cursar

Pertenece

Al transformar la relación de uno a muchos (1:N) `Pertenece` (Figura 2.17) al Modelo Relacional, se aplica la regla de propagación de claves. El procedimiento es el siguiente:

- La clave primaria de la entidad en el lado "uno" de la relación (`Departamento`) se propaga a la tabla de la entidad en el lado "muchos" (`Alumno`).
- La clave primaria de `Departamento` (`Numero`), se añade como clave foránea a la tabla `Alumno`.
- Como la participación del `Alumno` es obligatoria (1,1), según se infiere de la línea 13, estas nuevas columnas en la tabla `Alumno` no deben permitir valores nulos.

La estructura resultante de la tabla `Alumno` se detalla en la Tabla 2.9.

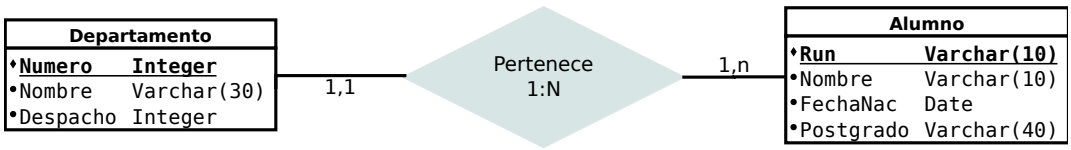

Figura 2.17: Relación Pertenece

Alumno	
Run (PK)	Varchar(10)
Nombre	Varchar(40)
Fechanac	Date
Postgrado_id (FK)	Integer
Departamento_id (FK)	Integer

Tabla 2.9: MR de la Relación Pertenece

2.3.4. Conversión Relaciones 1:1

Administrar

Para convertir la relación de uno a uno (1:1) `Administrar` (Figura 2.18) al Modelo Relacional, se utiliza el método de propagación de claves. La dirección de esta propagación se decide en función de la participación de cada entidad. Se propaga la clave primaria de la entidad con participación opcional a la tabla de la entidad con participación obligatoria.

- La entidad `Departamento` tiene participación obligatoria y única (1,1).
- La entidad `Profesor` tiene participación opcional y única (0,1).

Para optimizar el diseño, la clave primaria de `Profesor` se añade como clave foránea en la tabla `Departamento`. Para garantizar la integridad de la relación 1:1, esta nueva columna debe ser `NOT NULL` y `UNIQUE`. La estructura resultante se detalla en la Tabla 2.10.

Figura 2.18: Relación Administrar

Departamento	
Numero (PK)	Integer
Nombre	Varchar(30)
Despacho	Integer
Profesor_id (FUK)	Varchar(10)

Tabla 2.10: MR de la Relación Administrar

2.3.5. Conversión Relaciones Reflexivas y otras

Asesorar

A partir del requisito de la línea 14, se define la relación reflexiva (o recursiva) `Asesorar` sobre la propia entidad `Alumno`. Como se ilustra en la Figura 2.19, esta es una relación de uno a muchos (1:N) que distingue los roles de "asesor" y "asesorado":

- Un `Alumno` (en el rol de asesor) puede asesorar a uno o muchos otros alumnos (1,N).
- Cada `Alumno` (en el rol de asesorado) debe tener exactamente un asesor (1,1).

Para implementar esto en el Modelo Relacional, se añade una clave foránea en la tabla `Alumno` que se referencia a sí misma, como se muestra en la Tabla 2.11.

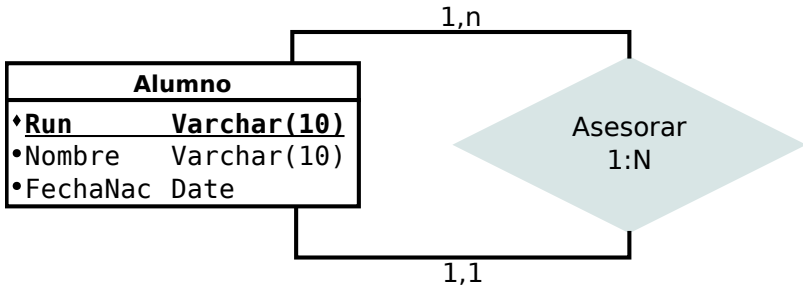

Figura 2.19: Relación Asesorar

Alumno	
Run (PK)	Varchar(10)
Nombre	Varchar(40)
Fechanac	Date
Postgrado_id (FK)	Integer
Departamento_id (FK)	Integer
Alumno_id (FK)	Varchar(10)

Tabla 2.11: MR de la Relación Asesorar

2.3.6. Modelo Relacional Resultante

Una vez realizadas las conversiones presentadas en las secciones 2.3.1-2.3.5 se obtienen las entidades que se ven en la Figura 2.20.

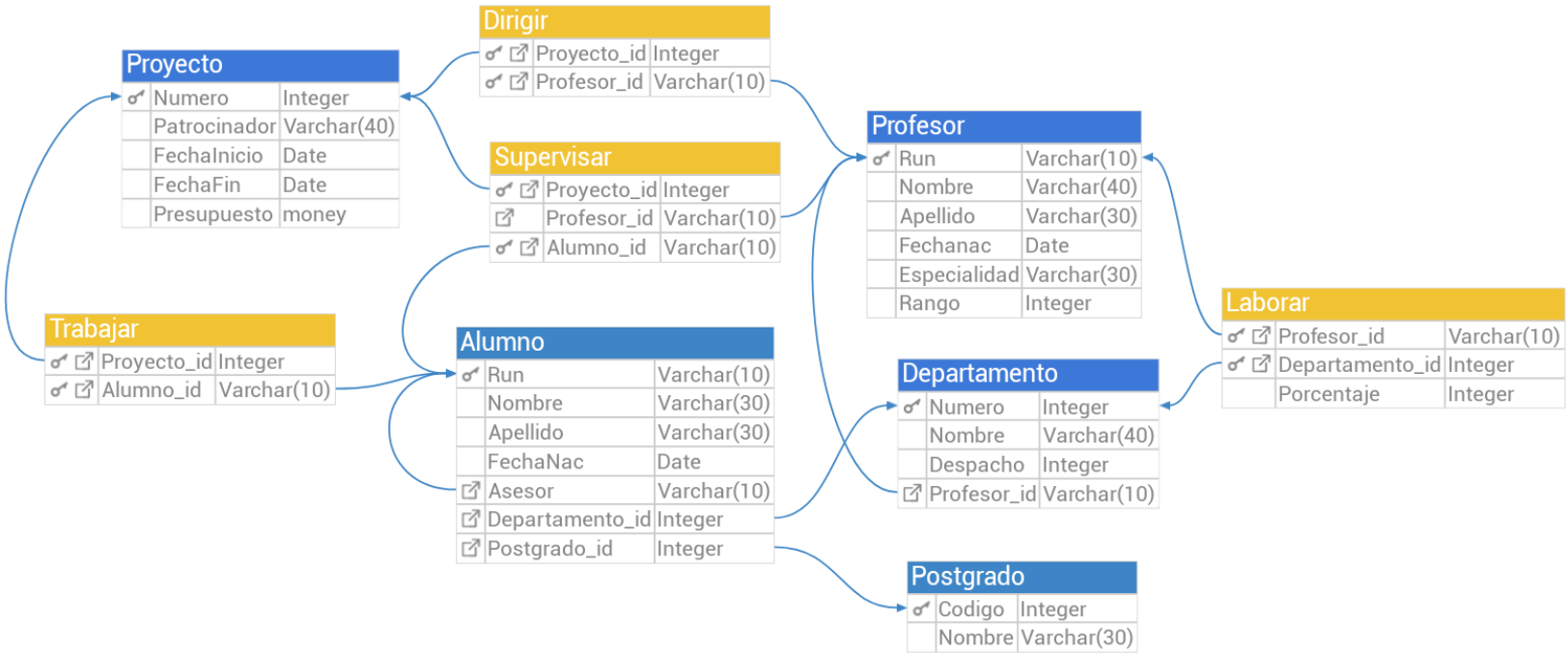

Figura 2.20: Modelo Relacional de la universidad

2.4. Normalización

A continuación, se detalla el análisis de normalización para cada tabla del esquema proporcionado, identificando si se encuentran en Tercera Forma Normal (3NF) o en una forma normal inferior. Se asume que todas las tablas ya cumplen con la Primera Forma Normal (1NF).

2.4.1. Tablas que Cumplen con la Tercera Forma Normal (3NF)

La gran mayoría de las tablas en el esquema proporcionado se encuentran en 3NF. Una tabla está en 3NF si ya está en 2NF y no existen dependencias transitivas (un atributo no-clave no depende de otro atributo no-clave).

- `Proyecto`:
  - **Clave Primaria (PK):** `Numero`.
  - **Análisis:** Al tener una clave primaria simple, la tabla está automáticamente en 2NF. No existen dependencias transitivas; los atributos `Patrocinador`, `FechaInicio`, `FechaFin` y `Presupuesto` dependen directamente del `Numero` del proyecto. **Se encuentra en 3NF**.
- `Postgrado`:
  - **PK:** `Codigo`.
  - **Análisis:** Tiene una clave simple y un único atributo no-clave (`Nombre`) que depende directamente de ella. **Se encuentra en 3NF**.
- `Profesor`:
  - **PK:** `Run`.
  - **Análisis:** Tiene una clave simple, por lo que cumple 2NF. No existen dependencias transitivas entre sus atributos no-clave como `Especialidad` o `Rango`. **Se encuentra en 3NF**.
- `Dirigir` y `Trabajar`:
  - **Análisis:** Son tablas asociativas puras que solo contienen las claves foráneas que forman su clave primaria. Al no tener atributos no-clave, no pueden tener dependencias parciales ni transitivas. **Se encuentran en 3NF** (y en una forma superior, BCNF).
- `Laborar`:
  - **PK:** `(Profesor_id, Departamento_id)`.

- **Análisis:** El único atributo no-clave, `Porcentaje`, depende de la combinación completa de un profesor y un departamento, no de solo uno de ellos, por lo que cumple con 2NF. Al no haber otros atributos no-clave, no pueden existir dependencias transitivas. **Se encuentra en 3NF**.

- `Alumno`:
  - **PK:** `Run`.
  - **Análisis:** Tiene clave simple, cumpliendo 2NF. Los atributos no-clave, incluyendo las claves foráneas `Asesor`, `Departamento_id` y `Postgrado_id`, son todos características directas del alumno y no dependen entre sí. **Se encuentra en 3NF**.

- `Supervisar`:
  - **PK:** `(Proyecto_id, Alumno_id)`.
  - **Análisis:** El atributo no-clave `Profesor_id` depende funcionalmente de la clave primaria completa, ya que un alumno en un proyecto específico tiene un único supervisor. Por lo tanto, cumple con 2NF. Al no haber otros atributos no-clave, no es posible una dependencia transitiva. **Se encuentra en 3NF**.

2.4.2. Tablas que NO se Encuentran en Tercera Forma Normal (3NF)

Basado en una interpretación lógica de los atributos, solo una tabla presenta una clara violación a la 3NF.

- `Departamento`:
  - **PK:** `Numero`.
  - **Atributos no-clave:** `Nombre`, `Despacho`, `Profesor_id`.
  - **Análisis de Violación a 3NF:**
    1. La tabla tiene una clave primaria simple (`Numero`), por lo que cumple con la 2NF.
    2. Sin embargo, existe una **dependencia transitiva** si se asume que el `Despacho` (oficina) del departamento es en realidad la oficina personal del profesor que lo dirige (`Profesor_id`).
    3. Bajo esta suposición, tenemos la siguiente cadena de dependencias funcionales:

$$\texttt{Numero} \rightarrow \texttt{Profesor\_id}$$

$$\texttt{Profesor\_id} \rightarrow \texttt{Despacho}$$

    4. Como el atributo no-clave `Despacho` depende de otro atributo no-clave (`Profesor_id`), y no directamente de la clave primaria `Numero`, la tabla `Departamento` **no se encuentra en 3NF**. Estaría en 2NF.

Proceso de Normalizar

2.4.3. Modelo Relacional Normalizado

Considerando las entidades que debieron ser normalizadas de la sección 2.4.2 se obtiene el conjunto de entidades completamente normalizada en 3ra Forma Normal de la Figura 2.21, este nivel de normalización asegura una estabilidad y rapidez en las consultas considerable.

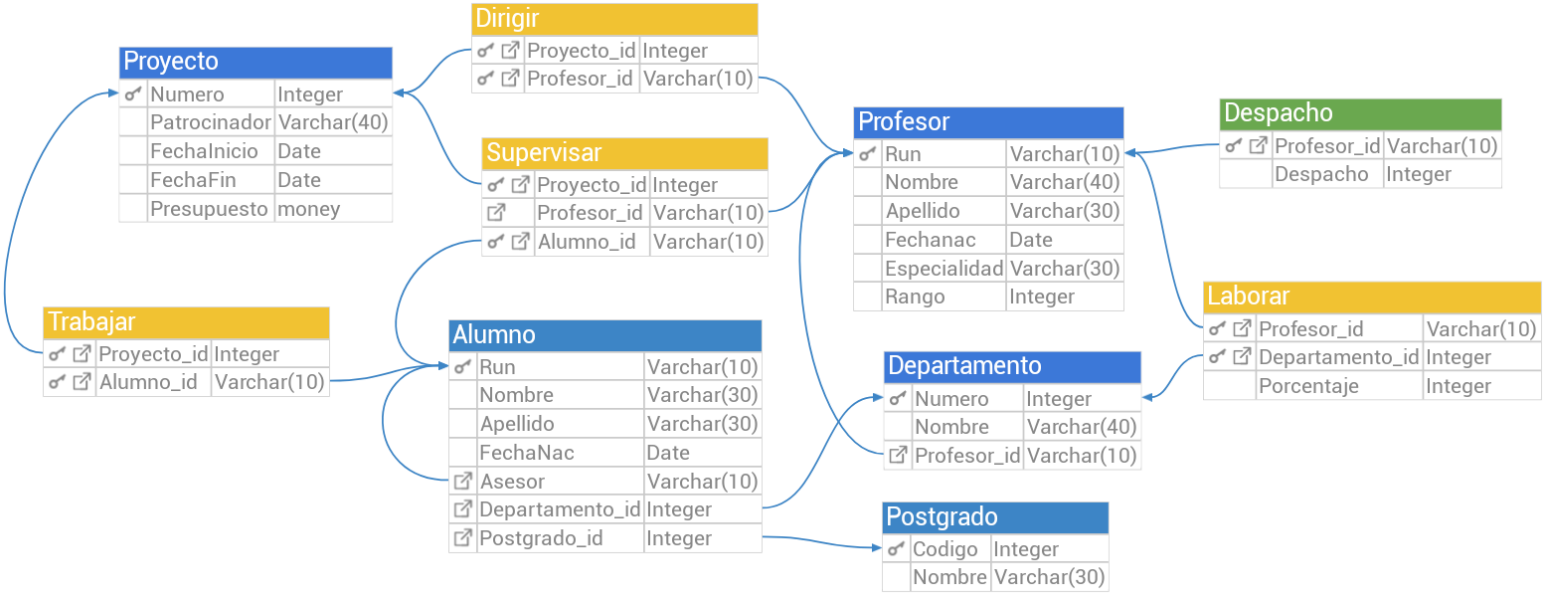

Figura 2.21: Modelo Relacional Normalizado de la universidad

Juaramir: Quiero aclarar que para el resto del informe se asumirá que el despacho es una oficina independiente del profesor que administra el departamento, por lo que no se considerará esta normalización en el resto del informe. **END.**

Capítulo 3

Lenguaje de Definición de Datos

3.1. Creación de Tablas

En el Código 3.1 se presentan las eliminaciones de las tablas en caso de ser necesario, en el Código 3.2 se presentan los `create table` del sistema, basados en el modelo relacional normalizado de la Figura 2.21.

Código 3.1: Eliminación de las Tablas

```
DROP TABLE IF EXISTS Proyecto;
DROP TABLE IF EXISTS Departamento;
DROP TABLE IF EXISTS Profesor;
DROP TABLE IF EXISTS Trabajar;
DROP TABLE IF EXISTS Supervisar;
DROP TABLE IF EXISTS Dirigir;
DROP TABLE IF EXISTS Laborar;
DROP TABLE IF EXISTS Alumno;
DROP TABLE IF EXISTS Postgrado;
```

Código 3.2: Creaciones de Tablas

```
CREATE TABLE Proyecto (
Numero INTEGER PRIMARY KEY NOT NULL,
Patrocinador VARCHAR(40),
FechaInicio DATE NOT NULL,
FechaFin DATE,
Presupuesto MONEY);

CREATE TABLE Postgrado (
Codigo INTEGER PRIMARY KEY NOT NULL,
Nombre VARCHAR(30) NOT NULL);

```

```
12 CREATE TABLE Profesor (
13 Run VARCHAR(10) PRIMARY KEY NOT NULL,
14 Nombre VARCHAR(40),
15 Apellido VARCHAR(30),
16 Fechanac DATE,
17 Especialidad VARCHAR(30),
18 Rango INTEGER);
19
20 CREATE TABLE Departamento (
21 Numero INTEGER PRIMARY KEY NOT NULL,
22 Nombre VARCHAR(40),
23 Despacho INTEGER,
24 Profesor_id VARCHAR(10) NOT NULL UNIQUE,
25 FOREIGN KEY (Profesor_id) REFERENCES Profesor(Run) ON DELETE ↩
      CASCADE ON UPDATE CASCADE);
26
27 CREATE TABLE Dirigir (
28 Proyecto_id INTEGER NOT NULL,
29 Profesor_id VARCHAR(10) NOT NULL,
30 FOREIGN KEY (Proyecto_id) REFERENCES Proyecto(Numero) ON DELETE ↩
      CASCADE ON UPDATE CASCADE,
31 FOREIGN KEY (Profesor_id) REFERENCES Profesor(Run) ON DELETE ↩
      CASCADE ON UPDATE CASCADE,
32 PRIMARY KEY (Proyecto_id,Profesor_id));
33
34 CREATE TABLE Laborar (
35 Profesor_id VARCHAR(10) NOT NULL,
36 Departamento_id INTEGER NOT NULL,
37 Porcentaje INTEGER NOT NULL,
38 FOREIGN KEY (Profesor_id) REFERENCES Profesor(Run) ON DELETE ↩
      CASCADE ON UPDATE CASCADE,
39 FOREIGN KEY (Departamento_id) REFERENCES Departamento(Numero) ON ↩
      DELETE CASCADE ON UPDATE CASCADE
40 PRIMARY KEY (Profesor_id,Departamento_id));
41
42 CREATE TABLE Alumno (
43 Run VARCHAR(10) PRIMARY KEY NOT NULL,
44 Nombre VARCHAR(30) NOT NULL,
45 Apellido VARCHAR(30) NOT NULL,
46 FechaNac DATE NOT NULL,
47 Asesor VARCHAR(10) NOT NULL,
48 Departamento_id INTEGER NOT NULL,
```

```
49 Postgrado_id INTEGER,
50 FOREIGN KEY (Asesor) REFERENCES Alumno(Run) ON DELETE CASCADE ON ↩
      UPDATE CASCADE,
51 FOREIGN KEY (Departamento_id) REFERENCES Departamento(Numero) ON ↩
      DELETE CASCADE ON UPDATE CASCADE,
52 FOREIGN KEY (Postgrado_id) REFERENCES Postgrado(Codigo) ON DELETE↩
       SET NULL ON UPDATE CASCADE);
53
54 CREATE TABLE Supervisar (
55 Proyecto_id INTEGER NOT NULL,
56 Profesor_id VARCHAR(10) NOT NULL,
57 Alumno_id VARCHAR(10) NOT NULL,
58 FOREIGN KEY (Proyecto_id) REFERENCES Proyecto(Numero) ON DELETE ↩
      CASCADE ON UPDATE CASCADE,
59 FOREIGN KEY (Profesor_id) REFERENCES Profesor(Run) ON DELETE ↩
      CASCADE ON UPDATE CASCADE,
60 FOREIGN KEY (Alumno_id) REFERENCES Alumno(Run) ON DELETE CASCADE ↩
      ON UPDATE CASCADE,
61 PRIMARY KEY (Proyecto_id,Alumno_id));
62
63 CREATE TABLE Trabajar (
64 Proyecto_id INTEGER NOT NULL,
65 Alumno_id VARCHAR(10) NOT NULL,
66 FOREIGN KEY (Proyecto_id) REFERENCES Proyecto(Numero) ON DELETE ↩
      CASCADE ON UPDATE CASCADE,
67 FOREIGN KEY (Alumno_id) REFERENCES Alumno(Run) ON DELETE CASCADE ↩
      ON UPDATE CASCADE,
68 PRIMARY KEY (Proyecto_id,Alumno_id));
```

3.2. Inserción de datos y demostración

A continuación se presentan las sentencias SQL `INSERT` para poblar las tablas de la base de datos creada con el Código 3.2, solo se presentarán las 10 primeras tuplas, tanto para los insert como para mostrar la tabla cargada en la base de datos.

Código 3.3: Datos para la tabla Profesor

```
1 -- Rango: 1=Instructor, 2=Asistente, 3=Asociado, 4=Titular
2 INSERT INTO Profesor (Run, Nombre, Apellido, FechaNac, ↩
      Especialidad, Rango) VALUES
3 ('12345678-5', 'Ana', 'Pérez', '1975-03-12', 'Bases de Datos', 4)↩
      ,
```

```
('11222333-K', 'Luis', 'Gómez', '1980-11-25', 'IA', 3),
('98765432-1', 'Carla', 'Morales', '1985-01-18', 'Ing. de Software', 2),
('87654321-0', 'Juan', 'Soto', '1972-07-30', 'Sist. Operativos', 4),
('76543210-K', 'Sofía', 'Castro', '1990-09-01', 'Redes y Seguridad', 2),
('15888777-3', 'Miguel', 'Rojas', '1968-04-15', 'Compiladores', 4),
('13456789-0', 'Elena', 'Vargas', '1982-06-20', 'Computación Gráfica', 3),
('14258369-7', 'Javier', 'Reyes', '1988-12-05', 'HCI', 2),
('10987654-3', 'Lucía', 'Méndez', '1979-10-10', 'Algoritmos', 3),
('16123456-7', 'Pedro', 'Fuentes', '1984-02-28', 'Sist. Distribuidos', 2);
```

Juaramir: Aquí poner Imagen de la tabla con datos en el motor **END.**

Código 3.4: Datos para la tabla Postgrado

```
INSERT INTO Postgrado (Codigo, Nombre) VALUES
(100, 'Magíster en Cs. Computación'),
(101, 'Magíster en Ing. de Software'),
(200, 'Doctorado en Cs. Computación'),
(201, 'Doctorado en IA'),
(300, 'Magíster en Data Science'),
(301, 'Magíster en Ciberseguridad'),
(400, 'Postítulo en Redes'),
(401, 'Postítulo en Desarrollo Web'),
(102, 'Magíster en Bioinformática'),
(202, 'Doctorado en Robótica'),
(302, 'Magíster en Videojuegos');
```

Juaramir: Aquí poner Imagen de la tabla con datos en el motor **END.**

Código 3.5: Datos para la tabla Proyecto

```
INSERT INTO Proyecto (Numero, Patrocinador, FechaInicio, FechaFin, Presupuesto) VALUES
(10, 'ANID-FONDECYT', '2023-03-15', '2026-03-14', 120000000),
(11, 'CORFO', '2024-01-20', '2025-07-20', 75000000),
(12, 'Interno U.', '2024-08-01', '2025-02-01', 30000000),
(13, 'Microsoft Research', '2023-09-01', NULL, 250000000),
(14, 'Google AI', '2024-06-01', '2026-05-31', 180000000),
```

```
(15, 'ANID-FONDEF', '2025-03-01', '2027-02-28', 95000000),
(16, 'Innova Chile', '2023-11-10', NULL, 60000000),
(17, 'CERN', '2024-05-01', '2026-04-30', 220000000),
(18, 'Tesla', '2025-01-01', NULL, 300000000),
(19, 'Interno LabComp', '2024-09-01', '2025-03-01', 25000000);
```

Juaramir: Aquí poner Imagen de la tabla con datos en el motor **END.**

Código 3.6: Datos para la tabla Departamento

```
INSERT INTO Departamento (Numero, Nombre, Despacho, Profesor_id) VALUES
(1, 'Ingeniería de Software', 301, '98765432-1'),
(2, 'Ciencias de la Computación', 210, '12345678-5'),
(3, 'Sistemas y Computación', 420, '87654321-0'),
(4, 'Redes y Ciberseguridad', 530, '76543210-K'),
(5, 'Inteligencia Artificial', 215, '11222333-K'),
(6, 'Computación Gráfica y Robótica', 110, '13456789-0'),
(7, 'Algoritmia y Teoría', 222, '10987654-3'),
(8, 'Sistemas Distribuidos', 333, '16123456-7'),
(9, 'Interacción Humano-Computador', 123, '14258369-7'),
(10, 'Centro de Investigación', 555, '15888777-3');
```

Juaramir: Aquí poner Imagen de la tabla con datos en el motor **END.**

Código 3.7: Datos para la tabla Alumno

```
-- El Asesor es NOT NULL, por lo que se crea una cadena de asesoría
INSERT INTO Alumno (Run, Nombre, Apellido, FechaNac, Asesor, Departamento_id, Postgrado_id) VALUES
('20111222-3', 'Fernanda', 'Rojas', '2001-05-20', '20333444-5', 2, 100),
('20222333-4', 'Martín', 'Soto', '2002-02-15', '20111222-3', 1, 101),
('20333444-5', 'Camila', 'Vargas', '2000-10-10', '20222333-4', 2, 200),
('19888777-6', 'Diego', 'Castro', '1999-12-01', '20333444-5', 4, 301),
('19777666-5', 'Valentina', 'Morales', '1998-03-25', '20111222-3', 3, NULL),
('21123456-7', 'Benjamín', 'Herrera', '2003-01-30', '20222333-4', 5, 201),
('21543210-9', 'Isidora', 'Jiménez', '2002-08-14', '19888777-6',
```

```
   7, NULL),
('19654321-8', 'Tomás', 'Flores', '1999-07-07', '19777666-5', 6, 102),
('20876543-2', 'Antonia', 'Reyes', '2001-04-18', '21123456-7', 8, 300),
('21001002-K', 'Sebastián', 'Silva', '2003-06-22', '19654321-8', 9, 202),
('19555444-3', 'Javiera', 'Lagos', '1998-11-11', '20111222-3', 2, NULL),
('19444333-2', 'Matías', 'Nuñez', '1999-09-09', '20333444-5', 1, 101);
```

Juaramir: Aquí poner Imagen de la tabla con datos en el motor **END.**

Código 3.8: Datos para la tabla Dirigir

```
INSERT INTO Dirigir (Proyecto_id, Profesor_id) VALUES
(10, '12345678-5'), -- Ana dirige proyecto 10
(11, '98765432-1'), -- Carla dirige proyecto 11
(12, '98765432-1'), -- Carla también dirige proyecto 12
(13, '11222333-K'), -- Luis dirige proyecto 13
(14, '11222333-K'), -- Luis también dirige proyecto 14
(15, '76543210-K'), -- Sofía dirige proyecto 15
(16, '87654321-0'), -- Juan dirige proyecto 16
(17, '15888777-3'), -- Miguel dirige proyecto 17
(18, '13456789-0'), -- Elena dirige proyecto 18
(19, '14258369-7'); -- Javier dirige proyecto 19
```

Juaramir: Aquí poner Imagen de la tabla con datos en el motor **END.**

Código 3.9: Datos para la tabla Laborar

```
INSERT INTO Laborar (Profesor_id, Departamento_id, Porcentaje) VALUES
('12345678-5', 2, 100), -- Ana en Cs. de la Computación
('11222333-K', 5, 100), -- Luis en IA
('98765432-1', 1, 100), -- Carla en Ing. de Software
('87654321-0', 3, 100), -- Juan en Sistemas y Computación
('76543210-K', 4, 100), -- Sofía en Redes y Ciberseguridad
('15888777-3', 2, 50),  -- Miguel 50% en Cs. de la Comp.
('15888777-3', 7, 50),  -- y 50% en Algoritmia
('13456789-0', 6, 100), -- Elena en Comp. Gráfica
('10987654-3', 7, 100), -- Lucía en Algoritmia
('16123456-7', 8, 100), -- Pedro en Sist. Distribuidos
```

```
('14258369-7', 9, 100); -- Javier en HCI
```

Juaramir: Aquí poner Imagen de la tabla con datos en el motor **END.**

Código 3.10: Datos para la tabla Trabajar

```
INSERT INTO Trabajar (Proyecto_id, Alumno_id) VALUES
(10, '20111222-3'), -- Fernanda en proyecto 10
(10, '20333444-5'), -- Camila en proyecto 10
(11, '20222333-4'), -- Martín en proyecto 11
(13, '21123456-7'), -- Benjamín en proyecto 13
(13, '20111222-3'), -- Fernanda también en proyecto 13
(14, '19888777-6'), -- Diego en proyecto 14
(15, '21543210-9'), -- Isidora en proyecto 15
(17, '19654321-8'), -- Tomás en proyecto 17
(18, '20876543-2'), -- Antonia en proyecto 18
(18, '21001002-K'), -- Sebastián en proyecto 18
(19, '19555444-3'); -- Javiera en proyecto 19
```

Juaramir: Aquí poner Imagen de la tabla con datos en el motor **END.**

Código 3.11: Datos para la tabla Supervisar

```
-- PK es (Proyecto_id, Alumno_id), lo que implica un único ↩
   supervisor por alumno en un proyecto
INSERT INTO Supervisar (Proyecto_id, Profesor_id, Alumno_id) ↩
   VALUES
(10, '12345678-5', '20111222-3'), -- En P10, Ana supervisa a ↩
   Fernanda
(10, '15888777-3', '20333444-5'), -- En P10, Miguel supervisa a ↩
   Camila
(11, '98765432-1', '20222333-4'), -- En P11, Carla supervisa a ↩
   Martín
(13, '11222333-K', '21123456-7'), -- En P13, Luis supervisa a ↩
   Benjamín
(13, '11222333-K', '20111222-3'), -- En P13, Luis también ↩
   supervisa a Fernanda
(14, '11222333-K', '19888777-6'), -- En P14, Luis supervisa a ↩
   Diego
(15, '76543210-K', '21543210-9'), -- En P15, Sofía supervisa a ↩
   Isidora
(17, '15888777-3', '19654321-8'), -- En P17, Miguel supervisa a ↩
   Tomás
(18, '13456789-0', '20876543-2'), -- En P18, Elena supervisa a ↩
```

```
    Antonia
(18, '13456789-0', '21001002-K'), -- En P18, Elena también supervisa a Sebastián
(19, '14258369-7', '19555444-3'); -- En P19, Javier supervisa a Javiera
```

Código 3.12: Datos insertados en la tabla Supervisar

```
select * from supervisar limit 10;
 proyecto_id | profesor_id | alumno_id
-------------+-------------+-------------
          10 | 12345678-5  | 20111222-3
          10 | 15888777-3  | 20333444-5
          11 | 98765432-1  | 20222333-4
          13 | 11222333-K  | 21123456-7
          13 | 11222333-K  | 20111222-3
          14 | 11222333-K  | 19888777-6
          15 | 76543210-K  | 21543210-9
          17 | 15888777-3  | 19654321-8
          18 | 13456789-0  | 20876543-2
          18 | 13456789-0  | 21001002-K
```

Capítulo 4

Lenguaje de Manipulación de Datos

4.1. Restricciones

Fecha de Inicio del Proyecto La fecha de inicio del proyecto debe ser igual o posterior a la fecha y hora actual del servidor.

Código 4.1: Restricción Fecha Inicio Proyecto

```
1 ALTER TABLE Proyecto ADD CONSTRAINT ↩
    chk_proyecto_fechainicio_valida CHECK (FechaInicio >= ↩
    CURRENT_TIMESTAMP);
2 -- Nota: now() es una función tradicional de PostgreSQL, ↩
    CURRENT_TIMESTAMP es el estándar SQL.
```

Código 4.2: Insert Impedido por la restricción chk_proyecto_fechainicio_valida

```
1 INSERT INTO Proyecto (Numero, Patrocinador, FechaInicio, FechaFin↩
    , Presupuesto) VALUES
2 (10, 'ANID-FONDECYT', '2023-03-15', '2026-03-14', 120000000);
3 ERROR:  new row for relation "proyecto" violates check constraint↩
     "chk_proyecto_fechainicio_valida"
4 DETAIL:  Failing row contains (10, ANID-FONDECYT, 2023-03-15, ↩
    2026-03-14, $ 120.000.000,00).
```

Dado que se desean almacenar proyectos históricos, se eliminará la restricción `chk_proyecto_fechainicio_valida`, esto se realiza como se muestra en el Código 4.3.

Código 4.3: Eliminar Restricción Fecha Inicio Proyecto

```
1 ALTER TABLE Proyecto DROP CONSTRAINT ↩
    chk_proyecto_fechainicio_valida;
```

Presupuesto No Negativo El presupuesto asignado a un proyecto no puede ser un valor negativo.

Código 4.4: Restricción Presupuesto no negativo

```
ALTER TABLE Proyecto
ADD CONSTRAINT chk_proyecto_presupuesto CHECK (Presupuesto >= '0'::MONEY);
```

Coherencia de Fechas del Proyecto Si se especifica una fecha de finalización para un proyecto, esta debe ser posterior a su fecha de inicio.

Código 4.5: Coherencia de Fechas del Proyecto

```
ALTER TABLE Proyecto
ADD CONSTRAINT chk_proyecto_fechas CHECK (FechaFin IS NULL OR FechaFin > FechaInicio);
```

Fecha de Nacimiento Válida La fecha de nacimiento de un profesor debe ser anterior a la fecha actual.

Código 4.6: Fecha de Nacimiento Válida

```
ALTER TABLE Profesor
ADD CONSTRAINT chk_profesor_fechanac CHECK (FechaNac < CURRENT_DATE);
```

Rango del Profesor El rango académico de un profesor debe pertenecer a una lista predefinida de valores válidos (ej. 1:'Titular', 2:'Asociado', 3:'Asistente', 4:'Instructor').

Código 4.7: Rango del Profesor

```
ALTER TABLE Profesor
ADD CONSTRAINT chk_profesor_rango CHECK (Rango BETWEEN 1 AND 4);
```

Auto-Asesoría Inválida Un alumno no puede ser registrado como su propio asesor.

Código 4.8: Auto-Asesoría Inválida

```
ALTER TABLE Alumno
ADD CONSTRAINT chk_alumno_no_autoasesor CHECK (Run != Asesor);
```

Porcentaje de Dedicación Válido El porcentaje de tiempo que un profesor labora en un departamento debe ser un valor positivo y no exceder el 100 %.

Código 4.9: Porcentaje de Dedicación Válido

```
ALTER TABLE Laborar
ADD CONSTRAINT chk_laborar_porcentaje CHECK (Porcentaje > 0 AND Porcentaje <= 100);
```

Número de departamento positivo

Código 4.10: Número de departamento positivo

```
ALTER TABLE Departamento
ADD CONSTRAINT chk_departamento_numero CHECK (Numero > 0);
```

Número de despacho positivo

Código 4.11: Número de despacho positivo

```
ALTER TABLE Departamento
ADD CONSTRAINT chk_departamento_despacho CHECK (Despacho > 0);
```

Código positivo

Código 4.12: Código positivo

```
ALTER TABLE Postgrado
ADD CONSTRAINT chk_postgrado_codigo CHECK (Codigo > 0);
```

4.2. Consultas y Vistas

Profesores de máximo rango Selecciona el nombre y la especialidad de todos los profesores que tienen el rango más alto (4).

Código 4.13: Profesores de máximo rango

```
SELECT Nombre, Apellido, Especialidad
FROM Profesor
WHERE Rango = 4;
```

Código 4.14: Resultado Código 4.13

```
 nombre | apellido |    especialidad
--------+----------+--------------------
 Ana    | Pérez    | Bases de Datos
 Juan   | Soto     | Sist. Operativos
 Miguel | Rojas    | Compiladores
```

Proyectos sin fecha de fin Obtiene el número, patrocinador y presupuesto de los proyectos que aún no tienen una fecha de finalización definida.

Código 4.15: Proyectos sin fecha de fin

```
SELECT   Numero,  Patrocinador,  Presupuesto
FROM Proyecto
```

```
WHERE  FechaFin IS NULL;
```

Código 4.16: Resultado Código 4.15

```
 numero |     patrocinador     |    presupuesto
--------+----------------------+--------------------
     13 | Microsoft Research   | $ 250.000.000,00
     16 | Innova Chile         |  $ 60.000.000,00
     18 | Tesla                | $ 300.000.000,00
```

Directores de Departamento (Join Implícito) Muestra el nombre de cada departamento y el nombre del profesor que lo dirige.

Código 4.17: Directores de Departamento (Join Implícito)

```
SELECT
  D.Nombre AS NombreDepartamento,
  P.Nombre AS NombreDirector,
  P.Apellido AS ApellidoDirector
FROM Departamento AS D, Profesor AS P
WHERE D.Profesor_id = P.Run;
```

Proyectos y sus Directores (INNER JOIN) Lista todos los proyectos junto con el nombre completo del profesor que los dirige.

Código 4.18: Proyectos y sus Directores (INNER JOIN)

```
SELECT
  Proy.Numero AS ProyectoID,
  Proy.Patrocinador,
  Prof.Nombre AS Director,
  Prof.Apellido AS ApellidoDirector
FROM Proyecto AS Proy
INNER JOIN Dirigir AS D
  ON Proy.Numero = D.Proyecto_id
INNER JOIN Profesor AS Prof
  ON D.Profesor_id = Prof.Run;
```

Presupuesto total por patrocinador Calcula la suma total de los presupuestos de los proyectos, agrupados por cada patrocinador.

Código 4.19: Presupuesto total por patrocinador

```
SELECT Patrocinador,
  SUM(Presupuesto) AS PresupuestoTotal
FROM Proyecto
```

```
GROUP BY Patrocinador
ORDER BY PresupuestoTotal DESC;
```

Profesores que dirigen más de un proyecto Utiliza `COUNT` para contar los proyectos por profesor y `HAVING` para filtrar y mostrar solo a aquellos que dirigen más de uno.

Código 4.20: Profesores que dirigen más de un proyecto

```
SELECT P.Nombre, P.Apellido,
  COUNT(D.Proyecto_id) AS ProyectosDirigidos
FROM Profesor AS P
INNER JOIN Dirigir AS D
  ON P.Run = D.Profesor_id
GROUP BY P.Run, P.Nombre, P.Apellido
HAVING COUNT(D.Proyecto_id) > 1;
```

Alumnos en programas de Magíster (Uso de IN) Encuentra a todos los alumnos cuyos programas de postgrado contienen la palabra 'Magíster'.

Código 4.21: Alumnos en programas de Magíster (Uso de IN)

```
SELECT Nombre, Apellido
FROM Alumno
WHERE
  Postgrado_id IN (
    SELECT Codigo
    FROM Postgrado
    WHERE Nombre LIKE 'Magíster%'
  );
```

Departamentos con profesores que trabajan al 100 % (Uso de EXISTS) Lista los departamentos donde existe al menos un profesor que dedica el 100 % de su tiempo a ese departamento.

Código 4.22: Departamentos con profesores que trabajan al 100 % (Uso de EXISTS)

```
SELECT  Nombre
FROM Departamento AS D
WHERE EXISTS (
    SELECT 1
    FROM Laborar AS L
    WHERE L.Departamento_id = D.Numero AND L.Porcentaje = 100
  );
```

Alumnos sin proyecto (Usando LEFT JOIN) La forma más común y eficiente. Se traen todos los alumnos y se unen con la tabla `Trabajar`. Aquellos que no tienen correspondencia (`t.Proyecto_id` es NULL) son los que no trabajan en proyectos.

Código 4.23: Alumnos sin proyecto (Usando LEFT JOIN)

```
SELECT A.Run, A.Nombre, A.Apellido
FROM Alumno AS A
LEFT JOIN Trabajar AS T
  ON A.Run = T.Alumno_id
WHERE T.Proyecto_id IS NULL;
```

Alumnos sin proyecto (Usando NOT EXISTS) Selecciona a los alumnos para los cuales no existe una entrada en la tabla `Trabajar` que los vincule a un proyecto.

Código 4.24: Alumnos sin proyecto (Usando NOT EXISTS)

```
SELECT  A.Nombre,  A.Apellido
FROM Alumno AS A
WHERE NOT EXISTS (
    SELECT 1
    FROM Trabajar AS T
    WHERE T.Alumno_id = A.Run
  );
```

Creación de una vista ProyectoDetallado Crea una vista que simplifica las consultas futuras al unir la información clave de los proyectos: el proyecto en sí, el director y el departamento del director.

Código 4.25: Creación de una vista ProyectoDetallado

```
CREATE VIEW VistaProyectoDetallado AS
SELECT
  Proy.Numero AS ProyectoNumero,
  Proy.Patrocinador,
  Proy.Presupuesto,
  Prof.Nombre AS NombreDirector,
  Prof.Apellido AS ApellidoDirector,
  Dep.Nombre AS DepartamentoDirector
FROM Proyecto AS Proy
INNER JOIN Dirigir AS Dir
  ON Proy.Numero = Dir.Proyecto_id
INNER JOIN Profesor AS Prof
  ON Dir.Profesor_id = Prof.Run
INNER JOIN Departamento AS Dep
```

```
15    ON Prof.Run = Dep.Profesor_id;
```

Uso de la vista para filtrar proyectos Una vez creada la vista, se puede consultar como si fuera una tabla normal para encontrar, por ejemplo, todos los proyectos dirigidos por un profesor del departamento de 'Inteligencia Artificial'.

Código 4.26: Uso de la vista para filtrar proyectos

```
1 SELECT  *
2 FROM VistaProyectoDetallado
3 WHERE  DepartamentoDirector = 'Inteligencia Artificial';
```

Supervisión de Alumnos en Proyectos Muestra una lista detallada de qué alumno es supervisado por qué profesor y en qué proyecto, mostrando los nombres en lugar de los IDs.

Código 4.27: Supervisión de Alumnos en Proyectos

```
1  SELECT
2    P.Nombre AS NombreProyecto,
3    Al.Nombre AS NombreAlumno,
4    Al.Apellido AS ApellidoAlumno,
5    Prof.Nombre AS NombreSupervisor,
6    Prof.Apellido AS ApellidoSupervisor
7  FROM Supervisar AS S
8  INNER JOIN Proyecto AS P
9    ON S.Proyecto_id = P.Numero
10 INNER JOIN Alumno AS Al
11   ON S.Alumno_id = Al.Run
12 INNER JOIN Profesor AS Prof
13   ON S.Profesor_id = Prof.Run
14 ORDER BY NombreProyecto;
```

Postgrado con más alumnos Encuentra el programa de postgrado con el mayor número de alumnos inscritos usando una combinación de `GROUP BY`, `ORDER BY`, `LIMIT` y una subconsulta.

Código 4.28: Postgrado con más alumnos

```
1 SELECT PG.Nombre,
2   COUNT(A.Run) AS NumeroDeAlumnos
3 FROM Postgrado AS PG
4 INNER JOIN Alumno AS A
5   ON PG.Codigo = A.Postgrado_id
6 GROUP BY PG.Nombre
7 ORDER BY NumeroDeAlumnos DESC
```

```
8 LIMIT 1;
```