STACMO

SIMULADOR DE TIRO DE ARTÍLLERIA DE CAMPAÑA Y MORTEROS

Diseño de Base de Datos Relacional para el Sistema de Simulación Militar STACMO

Curso: Programación Avanzada de Bases de Datos

Docente: ArianaJudith Uribe Rojas

Alumno: Gino Paolo Sassarini Bazán

Ciclo: 2025-II

RCN: 3073

Introducción

A lo largo de más de veinte años de trayectoria como diseñador de videojuegos y artista digital, he dedicado mi vida a crear mundos virtuales, personajes y experiencias interactivas capaces de enseñar, entretener y emocionar. En este camino descubrí que las mismas tecnologías que dan vida a los videojuegos podían trascender el entretenimiento y convertirse en herramientas poderosas para la formación y el entrenamiento en contextos reales.

Es en este punto donde nace STACMO, el primer simulador de tiro de artillería de campaña y morteros desarrollado íntegramente en el Perú, fruto del talento nacional y de una alianza estratégica con la empresa que dirijo, 3S Design. STACMO no es solo un producto tecnológico; es la materialización de una visión: aprovechar la creatividad y la innovación que normalmente asociamos al diseño de videojuegos para atender una necesidad crítica de nuestra realidad: preparar a los soldados en un entorno seguro, controlado y altamente eficiente.

El sistema está compuesto por varios módulos que interactúan entre sí de manera coordinada:

El Observador Avanzado (OA), quien detecta el objetivo y transmite los datos de rumbo, distancia y ángulo de situación.

La Central de Tiro (CT), donde los alumnos calculan el comando de tiro (deriva, situación y alza) en base a la información del OA.

El Instructor, que supervisa todo el proceso, compara los cálculos de los alumnos con el comando de tiro computado por el sistema, y evalúa la precisión del entrenamiento.

Finalmente, las Piezas Virtuales de Artillería, que ejecutan los disparos en el entorno digital y permiten observar el impacto de las decisiones tomadas.

Entorno del simulador STACMO, equipado con pantalla de gran formato, observación avanzada y puesto de control para el desarrollo de entrenamientos de artillería.

Representación 3D del diseño de espacios de entrenamiento del simulador STACMO, que integra aulas, central de tiro y observadores en un entorno coordinado.

Oficiales del Ejército del Perú interactuando con el sistema STACMO durante una sesión práctica de entrenamiento digital.

El valor de STACMO no radica únicamente en la tecnología de simulación que lo respalda, sino en la transformación que propone: reducir significativamente los costos y riesgos asociados al entrenamiento tradicional con munición real, al mismo tiempo que se eleva la calidad del aprendizaje gracias a la trazabilidad de cada acción dentro del simulador. Cada dato se registra, cada cálculo se conserva, y cada decisión puede analizarse para retroalimentar y fortalecer la instrucción.

En este sentido, STACMO se convierte en un laboratorio de aprendizaje, donde la teoría y la práctica convergen en un entorno digital que prepara mejor a quienes tienen la enorme responsabilidad de servir y defender al país.

Hoy, al cursar Programación Avanzada de Bases de Datos, me propongo dar un paso más: llevar a STACMO a un nivel superior. Los conocimientos que adquiera aquí no solo reforzarán mi formación como profesional, sino que también permitirán hacer más robusto el ecosistema digital del simulador.

Aspiro a que, a través de una mejor gestión de datos, STACMO sea capaz de ofrecer reportes más completos, estadísticas más precisas y una escalabilidad que lo convierta en un referente regional en simulación militar.

Este trabajo no es únicamente una práctica académica. Es, sobre todo, la oportunidad de hacer crecer un proyecto que ya lleva más de un año y medio de desarrollo y que se ha consolidado como un hito en la independencia tecnológica del Ejército del Perú. STACMO ha demostrado que el talento peruano es plenamente capaz de diseñar y ejecutar soluciones de simulación militar con el mismo nivel de calidad que las desarrolladas en cualquier otro país.

En ese sentido, el curso de Programación Avanzada de Bases de Datos se convierte en un pilar fundamental para seguir fortaleciendo este ecosistema: aplicar lo aprendido aquí no solo robustecerá la arquitectura de información de STACMO, sino que abrirá la puerta a nuevas funcionalidades, mayor escalabilidad y una gestión de datos más eficiente. Con ello, el simulador no solo cumple con su propósito inicial de entrenar en un entorno seguro y controlado, sino que también se convierte en un referente de cómo la academia y la industria pueden unirse para impulsar la innovación tecnológica en el país.

"De izquierda a derecha: Sergio Solano (Lead Developer), Tnte. Coronel Carlos Ruiz Aguirre (Responsable de Artillería del proyecto) y Gino Sassarini (CEO de 3S Design y Jefe del proyecto STACMO), durante una sesión de validación en el simulador."

Enunciado del Proyecto

El presente proyecto tiene como finalidad fortalecer la gestión de datos y optimizar la trazabilidad de la información generada por el Simulador de Tiro de Artillería de Campaña y Morteros (STACMO), actualmente en funcionamiento en la Escuela de Artillería del Ejército del Perú. STACMO, pionero en el país, ya reproduce con éxito los procesos de observación, cálculo y ejecución del tiro en un entorno digital seguro y controlado. No obstante, la evolución natural del sistema requiere de una arquitectura de datos más sólida y escalable, que permita registrar de manera estructurada cada observación, cálculo y resultado, facilitando un análisis más profundo del desempeño, la comparación entre cálculos manuales y computados, y la generación de reportes detallados que eleven la calidad del entrenamiento.

El simulador STACMO reproduce el ciclo completo de un tiro de artillería en un entorno digital y está compuesto por cuatro módulos principales:

Observador Avanzado (OA): registra rumbo, distancia y ángulo de situación de los objetivos.

Central de Tiro (CT): conformada por los alumnos, recibe los datos del OA y calcula el comando de tiro (deriva, situación y alza).

Instructor: supervisa el proceso, dispone del comando de tiro correcto generado automáticamente por el sistema y lo compara con los cálculos de los alumnos.

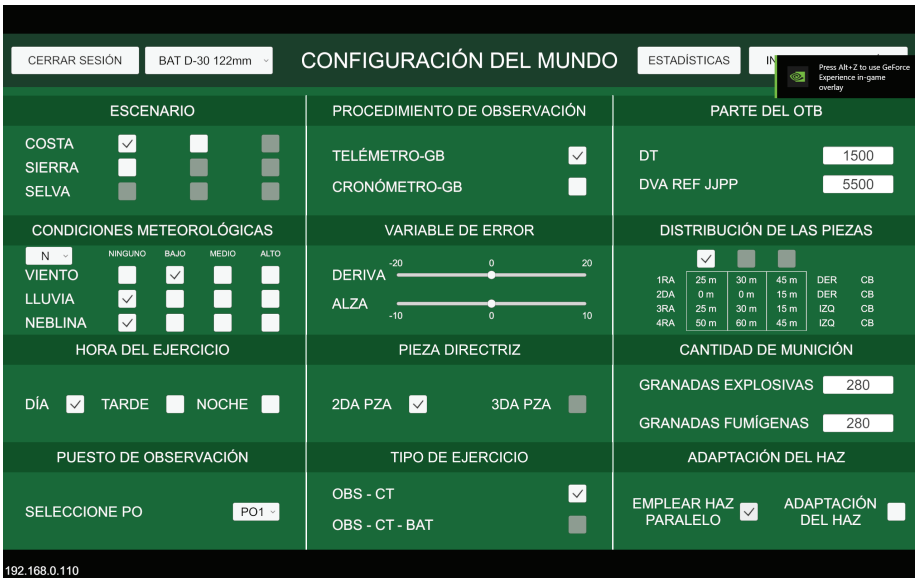
Piezas Virtuales de Artillería: representan obuses o morteros que ejecutan los disparos en el entorno digital y permiten visualizar los resultados de los cálculos.

Dentro de este ecosistema, cada usuario cumple un rol específico: los alumnos observan y miden como OA, practican sus cálculos en la CT, mientras que el instructor supervisa, evalúa y retroalimenta el desempeño. El sistema, por su parte, genera datos objetivos y matemáticamente correctos que sirven de referencia para medir la precisión de los cálculos manuales.

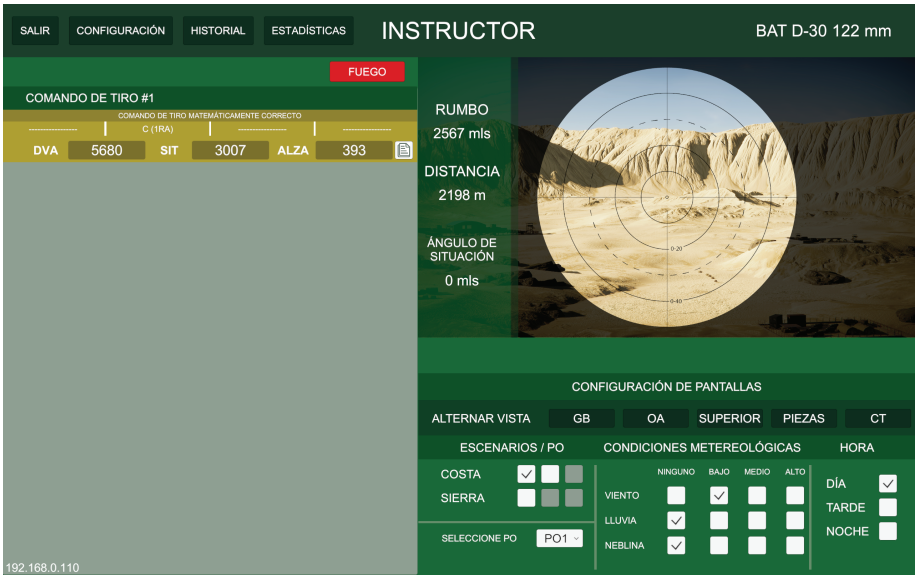
Los principales datos a gestionar son:

Observaciones del OA (rumbo, distancia, ángulo de situación).
Comandos de tiro de la CT (deriva, situación, alza).
Comando de tiro del sistema (referencia matemática).
Condiciones del ejercicio (escenario, clima, hora del día).
Munición y stock virtual consumido.
Calificaciones y métricas de desempeño individuales y grupales.

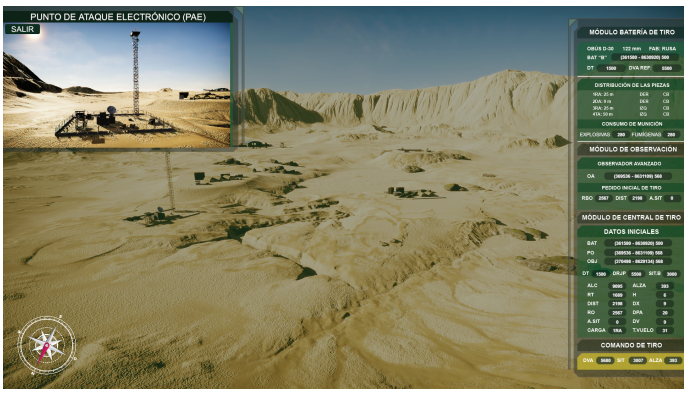
El desarrollo de un modelo de base de datos robusto y escalable es esencial para integrar toda esta información, garantizando su consistencia y validez, y facilitando la labor académica de la Escuela de Artillería. Con ello, se fortalece la capacidad de análisis, seguimiento y evaluación, consolidando a STACMO como una herramienta clave en la modernización del entrenamiento militar en el Perú.



Interfaz de configuración del simulador STACMO, que permite definir escenario, condiciones meteorológicas, variables técnicas, distribución de piezas y munición antes de iniciar un ejercicio de tiro.



Vista del módulo del Instructor en el simulador STACMO, donde se visualizan los parámetros del comando de tiro, las condiciones del ejercicio y la observación directa del objetivo en tiempo real.



Visualización del terreno desde la perspectiva del Observador Avanzado (OA), mostrando los objetivos disponibles y los datos técnicos asociados al ejercicio.

Vista a través del telémetro GB del Observador Avanzado, mostrando los datos de rumbo, distancia y ángulo de situación para la identificación precisa del objetivo.

Presentación de la Primera Entrega

El desarrollo del modelo conceptual del sistema STACMO constituye el primer paso hacia la construcción de una solución de base de datos robusta. En esta etapa, el objetivo principal fue abstraer la complejidad operativa del simulador y representarla en términos de entidades, atributos y relaciones, asegurando que la estructura de datos refleje fielmente los procesos de entrenamiento en artillería de campaña y morteros.

El trabajo inició con la identificación de las entidades principales que intervienen en el simulador: Alumno, Instructor, Aula, Grupo, GrupoMiembro, Ejercicio, ObservaciónOA, Objetivo, Ejercicio-Objetivo, ComandoCT, ComandoSist, Escenario, Condiciones, Munición y StockMunición. Estas entidades permiten capturar los diferentes actores y componentes del proceso de entrenamiento, desde la organización académica hasta la ejecución de un ejercicio y el consumo de recursos.

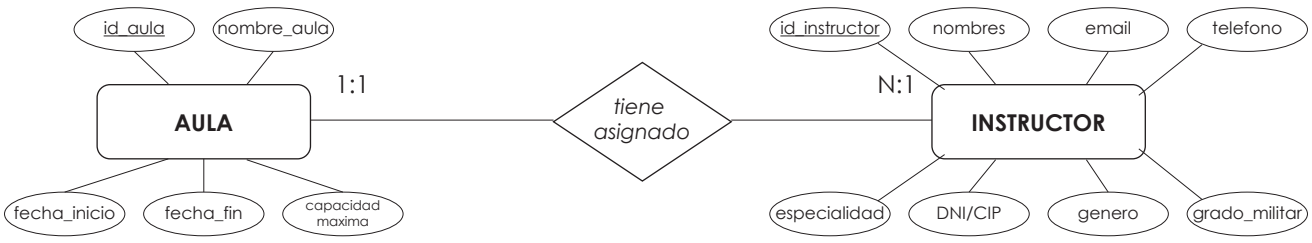
Cada entidad fue descrita con sus atributos más relevantes, incluyendo la definición de una clave primaria (PK) que asegura la identificación única de cada registro dentro del modelo. Este análisis detallado facilitó diferenciar las responsabilidades de cada entidad y evitar redundancias en la representación de la información.

Una vez establecidas las entidades y sus atributos, se procedió a definir las relaciones entre ellas, lo que resultó fundamental para representar cómo interactúan los distintos elementos dentro del simulador. Estas relaciones reflejan la dinámica del entrenamiento: los grupos formados en cada aula, las observaciones realizadas por el OA, los cálculos de la CT, los comandos generados por el sistema y el impacto de factores externos como el escenario, las condiciones ambientales y el stock de munición.

Posteriormente, se determinaron las cardinalidades de cada relación, aportando coherencia al modelo y asegurando que las conexiones entre entidades representaran de manera fiel la realidad del entrenamiento. Este paso permitió establecer, por ejemplo, que un instructor puede tener varias aulas asignadas, que un grupo está compuesto exactamente por un OA y tres CT, o que un ejercicio puede estar vinculado a múltiples objetivos.

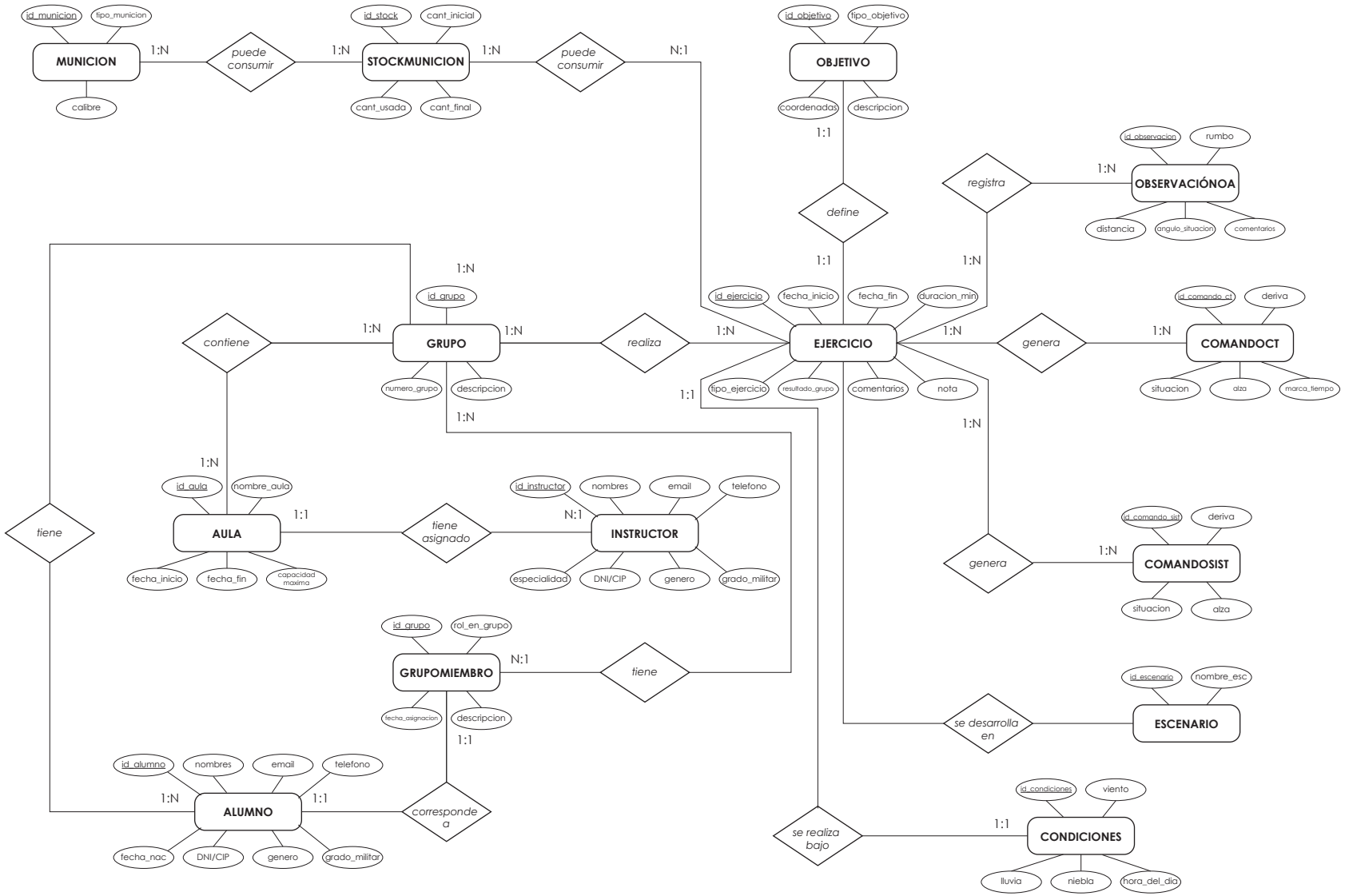
De forma complementaria, se elaboró una tabla resumen de cada entidad, en la que se consignaron sus atributos, la clave primaria y las relaciones más relevantes. Esta vista tabular no solo funcionó como un apoyo didáctico, sino que también facilitó la validación del modelo, al ofrecer una representación más sintética y comprensible que complementa la visión gráfica del diagrama conceptual.

El resultado es un modelo conceptual sólido, claro y validado, que organiza la información del simulador en términos de entidades y relaciones. Este esfuerzo no solo permitió documentar la estructura de datos actual de STACMO, sino también proyectar su crecimiento futuro en base a una arquitectura organizada. En conjunto, el diagrama y la tabla resumen sientan las bases para la siguiente etapa: el diseño del modelo lógico, en la que se incorporarán claves foráneas (FK), restricciones y reglas de integridad que garantizarán la consistencia de los datos y la trazabilidad completa dentro del sistema.



"Representación del diagrama conceptual del sistema STACMO, donde se muestran las entidades principales, sus atributos más relevantes y las cardinalidades que definen la relación entre ellas."

Diagrama conceptual



Explicación del Diagrama Conceptual

El diagrama conceptual del sistema STACMO organiza las entidades principales que intervienen en el proceso de entrenamiento de artillería y morteros, y las relaciones que existen entre ellas. El objetivo de este modelo es proporcionar una visión global y coherente de cómo se estructura la información, identificando a cada entidad con su llave primaria (PK) y señalando las llaves secundarias (FK) que permiten establecer los vínculos lógicos.

La explicación que se presenta a continuación detalla, entidad por entidad:

Sus atributos más relevantes.

La llave primaria que la identifica de manera única.

Las llaves secundarias que aseguran la correcta conexión con otras entidades.

La interpretación de las relaciones que se establecen en el diagrama, junto con sus cardinalidades.

De esta manera, el modelo no solo refleja la organización jerárquica de aulas, grupos, instructores y alumnos, sino también los elementos técnicos del simulador como ejercicios, observaciones, comandos, objetivos, escenarios, condiciones y stock de municiones. Con ello se asegura que cada aspecto del entrenamiento esté representado de forma clara y normalizada, sentando la base para posteriores etapas de diseño lógico y físico de la base de datos.

1. Entidad: Alumno

Representa a los estudiantes que participan en el simulador.

Atributos:
id_alumno (PK)
nombres, apellidos, email, DNI/CIP, género, grado_militar, fecha_nac, teléfono.

Relaciones:
Un Alumno corresponde a un GrupoMiembro (1:1).
Un Aula tiene muchos Alumnos (1:N).

2. Entidad: Instructor

Formador a cargo de un Aula.

Atributos:
id_instructor (PK)
nombres, apellidos, email, teléfono, grado_militar, género, especialidad, DNI/CIP.

Relación:
Cada Aula tiene asignado un solo Instructor (1:1).
Un Instructor puede estar asignado a varias Aulas (N:1).

3. Entidad: Aula

Agrupación académica donde se desarrollan los ejercicios.

Atributos:
id_aula (PK) ***id_instructor (FK)***
nombre_aula, fecha_inicio, fecha_fin, capacidad_maxima.

Relaciones:
Un Aula contiene varios Grupos (1:N).
Un Aula tiene varios Alumnos (1:N).
Un Aula tiene asignado un solo Instructor (1:1).

4. Entidad: Grupo

Subconjunto de alumnos dentro de un Aula.

Atributos:
id_grupo (PK) ***id_aula(FK)***
numero_grupo, descripcion.

Relaciones:
Un Grupo pertenece a un Aula (N:1).
Un Grupo se compone de varios GrupoMiembro (1:N).
Un Grupo realiza varios Ejercicios (1:N).

5. Entidad: GrupoMiembro

Relación intermedia que define el rol de cada Alumno dentro del Grupo.

Atributos:
id_grupomiembro (PK) ***id_grupo (FK)*** ***id_alumno (FK)***
rol_en_grupo (OA o CT), fecha_asignacion, observaciones.

Relaciones:
Cada GrupoMiembro corresponde a un Alumno (N:1).
Un Grupo tiene varios GrupoMiembro (1:N).

6. Entidad: Ejercicio

Actividad práctica realizada por un Grupo en un Escenario bajo determinadas Condiciones.

Atributos:
id_ejercicio (PK) ***id_grupo(FK)*** ***id_objetivo(FK)*** ***id_condiciones (FK)*** ***id_escenario (FK)***
fecha_inicio, fecha_fin, duracion_min, tipo_ejercicio, estado, nota, comentarios.

Relaciones:
Un Grupo realiza muchos Ejercicios (1:N).
Un Ejercicio define un Objetivo (1:1).
Un Ejercicio registra una ObservaciónOA (1:1 o 1:N según modelo).
Un Ejercicio genera varios ComandoCT (1:N).
Un Ejercicio genera uno o varios ComandoSistema (1:1 o 1:N).
Un Ejercicio se desarrolla en un Escenario (1:1).
Un Ejercicio se realiza bajo unas Condiciones (1:1).
Un Ejercicio puede consumir varios StockMunicion (1:N).

7. Entidad: Objetivo

Blanco definido del ejercicio.

Atributos:
id_objetivo (PK)
tipo_objetivo, coordenadas, descripcion.

Relación: Cada Ejercicio define un único Objetivo (1:1).

8. Entidad: ObservaciónOA

Datos iniciales proporcionados por el Observador Avanzado.

Atributos:
id_observacion (PK) ***id_ejercicio (FK)***
rumbo, distancia, angulo_situacion, comentarios.

Relación: Cada Ejercicio registra una ObservaciónOA (1:1 o 1:N).

9. Entidad: ComandoCT

Comando calculado por la Central de Tiro (los alumnos).

Atributos:
id_comando_ct(PK) ***id_ejercicio (FK)***
deriva, situacion, alza, version, marca_tiempo, comentarios.

Relación: Un Ejercicio genera varios ComandoCT (1:N).

10. Entidad: ComandoSistema

Comando calculado automáticamente por el sistema.

Atributos:
id_comando_sist (PK) ***id_ejercicio (FK)***
deriva, situacion, alza, metodo, marca_tiempo, comentarios.

Relación: Un Ejercicio genera uno o varios ComandoSistema (1:1 o 1:N).

11. Entidad: Escenario

Contexto geográfico del simulador.

Atributos:
id_escenario (PK)
nombre_esc, descripcion, tipo_terreno, extension_km2, mapa_base.

Relación: Un Ejercicio se desarrolla en un Escenario (1:1).

12. Entidad: Condiciones

Estado ambiental del ejercicio.

Atributos:
id_condiciones (PK)
viento, lluvia, niebla, hora_del_dia.

Relación: Un Ejercicio se realiza bajo unas Condiciones (1:1).

13. Entidad: Municion

Tipos de municiones disponibles.

Atributos:
id_municion (PK)
tipo_municion, calibre, descripcion.

Relación: Una Municion puede estar en muchos StockMunicion (1:N).

14. Entidad: StockMunicion

Control del consumo de municiones en un ejercicio.

Atributos:
id_stock (PK) ***id_ejercicio (FK)*** ***id_municion (FK)***
cantidad_inicial, cantidad_usada, cantidad_final, fecha_registro, observaciones.

Relaciones:
Un Ejercicio puede consumir varios StockMunicion (1:N).
Cada StockMunicion corresponde a un único tipo de Municion (N:1).

El desarrollo del modelo conceptual de STACMO permitió dar un primer paso firme hacia la consolidación de una base de datos robusta y alineada a las necesidades del simulador. A través de este proceso, se logró abstraer la complejidad operativa del entrenamiento en artillería de campaña y morteros y traducirla en un esquema claro de entidades, atributos y relaciones, garantizando que los datos más relevantes del sistema puedan ser representados de manera ordenada y comprensible.

Entre las principales conclusiones alcanzadas se destacan:

La identificación de las entidades clave permitió capturar de manera integral tanto los actores humanos (alumnos, instructor, grupos) como los elementos técnicos del entrenamiento (observaciones, cálculos de la CT, comandos del sistema, objetivos, condiciones, escenarios y municiones).

La definición de atributos y claves primarias en cada entidad aseguró una visión unificada de los datos, evitando redundancias y facilitando el posterior paso hacia un modelo lógico.

El establecimiento de relaciones y cardinalidades aportó realismo al modelo, reflejando con precisión la interacción entre los distintos módulos del simulador.

La construcción de una tabla resumen complementaria permitió verificar la consistencia del diagrama y funcionó como herramienta de validación y comprensión del modelo.

Las lecciones aprendidas de esta etapa también resultan significativas:

El trabajo conceptual es esencial para comprender a fondo el dominio del problema antes de pasar a fases más técnicas.

Representar primero las relaciones a nivel conceptual evitará confusiones posteriores en la definición de claves foráneas y restricciones, asegurando coherencia en el modelo.

La necesidad de expresar de manera clara las cardinalidades reforzó la importancia de alinear los requerimientos operativos con las reglas de negocio reales de la Escuela de Artillería.

Finalmente, se comprobó que un modelo conceptual bien diseñado no solo es un insumo técnico, sino también una herramienta de comunicación, capaz de transmitir de forma simple y estructurada cómo funciona el sistema y qué datos son fundamentales en su operación.

En síntesis, esta etapa no solo permitió obtener un diagrama conceptual validado y consistente, sino que también sentó las bases metodológicas y de comprensión necesarias para avanzar hacia el modelo lógico, donde se profundizará en la defi nición de claves foráneas, restricciones y reglas de integridad.

Desarrollo del Modelo Lógico

Tras la construcción del modelo conceptual, se procedió a la segunda etapa: el desarrollo del modelo lógico de la base de datos de STACMO.

Mientras que el modelo conceptual permitió representar de manera abstracta las entidades, atributos y relaciones que intervienen en el simulador, el modelo lógico avanza hacia un nivel más técnico y estructurado, acercándose al diseño que finalmente se implementará en un sistema de gestión de bases de datos.

Este paso resulta fundamental porque traduce la visión general del sistema en un esquema concreto y verificable, donde cada elemento del modelo comienza a asumir la forma que tendrá en la práctica.

En esta fase se identificaron y documentaron los elementos clave del modelo lógico:

Tablas: derivadas de las entidades conceptuales, transformadas en estructuras formales de almacenamiento de datos que representan de manera explícita la organización de la información.

Claves primarias (PK): asignadas en cada tabla para asegurar la unicidad de los registros y evitar ambigüedades en la identificación de la información.

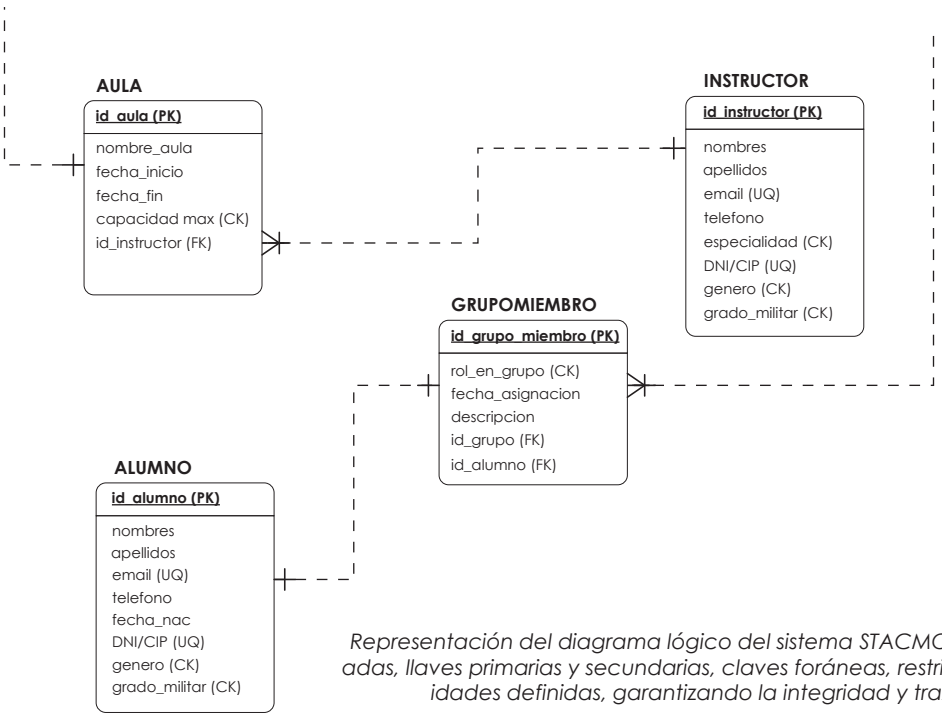
Relaciones: definidas a través de asociaciones entre tablas, reflejando las interacciones que existen entre los actores del simulador y las dinámicas del proceso de entrenamiento.

Claves foráneas (FK): incorporadas para establecer integridad referencial, garantizando que los vínculos entre tablas sean consistentes y que no existan registros huérfanos.

Restricciones (constraints): aplicadas para reflejar las reglas de negocio y las condiciones de validez que rigen el funcionamiento del sistema, incluyendo valores únicos, campos obligatorios o rangos permitidos en los datos.

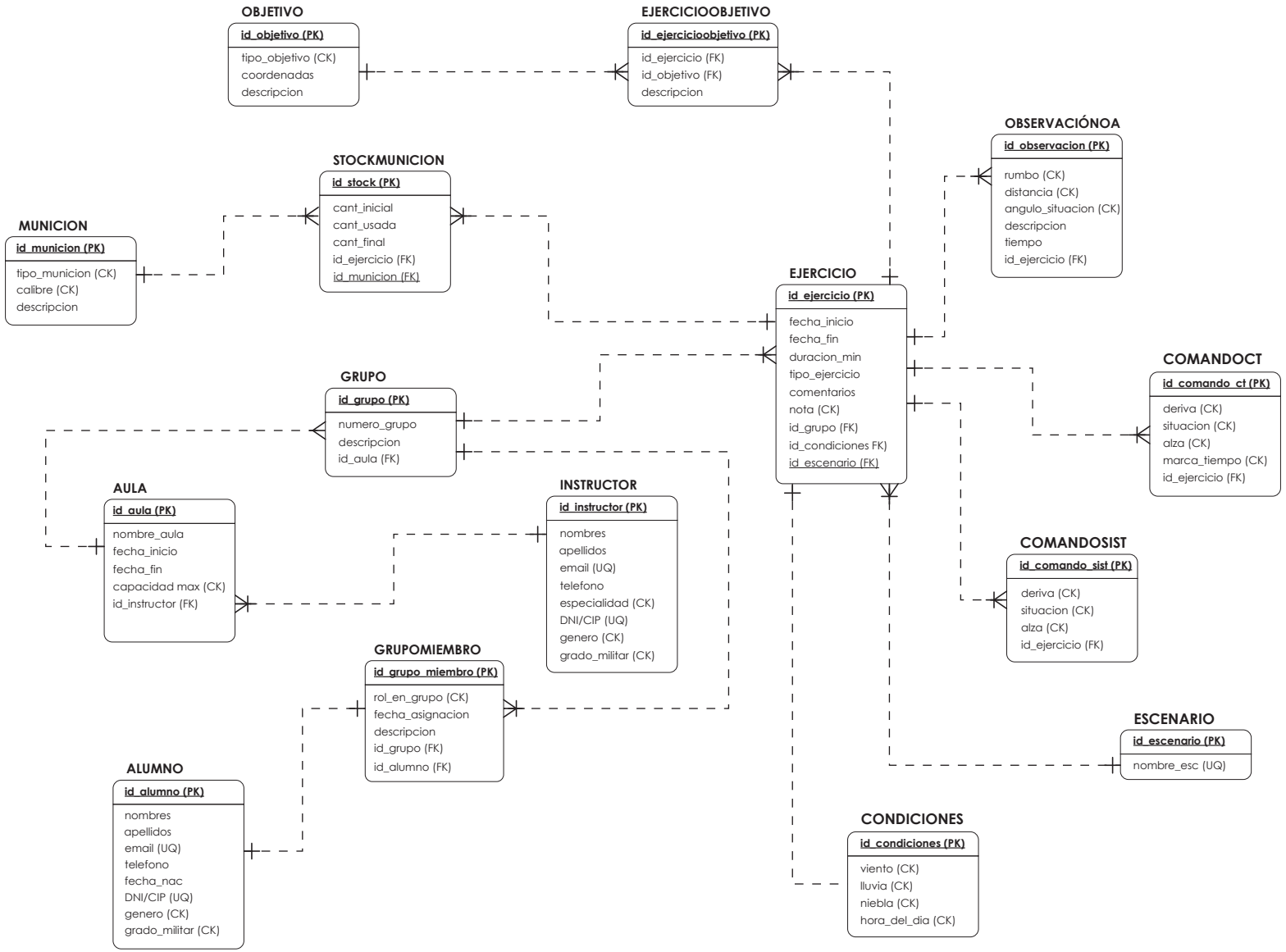
El trabajo de esta etapa no se limitó a la transformación mecánica del diagrama conceptual, sino que también implicó un proceso de análisis y refinamiento. Fue necesario revisar con detalle las relaciones y sus cardinalidades, definir qué atributos debían convertirse en claves foráneas, y establecer las restricciones necesarias para mantener la coherencia entre los datos y las reglas operativas de STACMO.

El resultado es un diagrama lógico claro, coherente y robusto, que no solo conserva la esencia del modelo conceptual inicial, sino que la lleva a un nivel de mayor formalidad técnica. Con este diseño, se asegura que la base de datos pueda responder adecuadamente a las necesidades de registro, trazabilidad y evaluación del simulador, garantizando al mismo tiempo la integridad y consistencia de la información.



Representación del diagrama lógico del sistema STACMO, con tablas estructuradas, llaves primarias y secundarias, claves foráneas, restricciones y las cardinalidades definidas, garantizando la integridad y trazabilidad de los datos.

Diagrama lógico



Explicación del Diagrama Conceptual

Tabla: INSTRUCTOR

Contiene los datos de los instructores que dirigen y supervisan los ejercicios dentro del simulador STACMO. Cada aula está obligatoriamente asignada a un instructor.

Atributo	Descripción	Constraints
id_instructor (PK)	Identificador único del instructor.	PK
nombres	Nombres completos del instructor.	
apellidos	Apellidos completos del instructor.	
email (UQ)	Dirección de correo electrónico institucional o personal.	
telefono	Número de contacto telefónico.	
especialidad (CK)	Área de especialización militar o técnica del instructor.	
DNI/CIP (UQ)	Documento nacional de identidad o carnet de identificación personal	
genero (CK)	Género del instructor.	CHECK (M/F)
grado_militar (CK)	Grado militar que posee el instructor (ej. Capitán, Mayor, etc.).	CHECK (Cadete, Alférez, Teniente, etc.)

Relaciones:

1 Instructor → N Aulas (un instructor puede estar asignado a varias aulas).
Cada aula tiene obligatoriamente 1 instructor.

Tabla: AULA

Representa el espacio académico en el que se organiza un conjunto de alumnos bajo la dirección de un instructor. Cada aula está compuesta por grupos de trabajo y constituye la unidad básica de organización para realizar los ejercicios en el simulador STACMO.

Atributo	Descripción	Constraints
id_aula	Identificador único del aula.	PK
id_instructor	Identificador del instructor asignado al aula.	FK
nombre_aula	Nombre o código identificador del aula.	
fecha_inicio	Fecha en que se abre o activa el aula.	
fecha_fin	Fecha en que se cierra el aula	
capacidad_max	Número máximo de alumnos permitidos en el aula.	CHECK (capacidad_max > 0)

Relaciones:

1 Instructor — N Aulas (cada aula pertenece a un solo instructor, pero un instructor puede manejar varias aulas).
1 Aula — N Grupos (cada aula contiene varios grupos de alumnos).
1 Aula — N Alumnos (indirecto vía GrupoMiembro).

Tabla: GRUPO

Subconjunto de alumnos dentro de un Aula con el que se ejecutan los Ejercicios. Cada grupo se identifica por un número que es único dentro de su aula (p. ej., 1, 2, 3). Un grupo está compuesto por 4 miembros (1 Observador Avanzado (OA) + 3 Miembros de la Central de Tiro (CT).

Atributo	Descripción	Constraints
id_grupo	Identificador único del grupo.	PK
id_aula	Aula a la que pertenece el grupo.	FK
numero_grupo	Número del grupo dentro del aula (1..3 o 1..5 según el caso).	
descripcion	Descripción breve del grupo (propósito, alumnos, etc).	

Relaciones:

1 Aula — N Grupos (cada aula puede contener varios grupos).
1 Grupo — N GrupoMiembro (cada grupo está compuesto por varios alumnos con roles definidos).
1 Grupo — N Ejercicios (un grupo puede realizar múltiples ejercicios a lo largo del tiempo).

Reglas

Cada grupo debe estar formado exactamente por 4 miembros.
Dentro del grupo debe haber 1 OA y 3 CT.
Un alumno no puede pertenecer a más de un grupo dentro de la misma aula.

Tabla: GRUPOMIEMBRO

Registra a los alumnos que integran un grupo dentro de un aula y el rol que cumplen en ese grupo (OA o CT). Es la tabla asociativa que materializa la relación Alumno–Grupo.

Atributo	Descripción	Constraints
id_grupo_miembro	Identificador único del registro de pertenencia de un alumno a un grupo	PK
id_grupo	Grupo al que pertenece el alumno.	FK
id_alumno	Alumno que integra el grupo.	FK
rol_en_grupo	Rol desempeñado en el grupo: OA (Observador Avanzado) o CT (Calculador de Tiro).	CHECK (OA, CT)
fecha_asignacion	Fecha en la que el alumno fue asignado al grupo.	
descripcion	Observaciones adicionales sobre la asignación.	

Relaciones:

1 Grupo — N GrupoMiembro (cada grupo se compone de varios miembros).

1 Alumno — N GrupoMiembro (un alumno puede tener varias asignaciones en el tiempo; en la misma aula).

Reglas

Por grupo debe haber exactamente 4 miembros: 1 OA + 3 CT.
Un alumno no puede pertenecer a más de un grupo dentro de la misma aula.

Tabla: ALUMNO

Almacena la información personal y académica de los alumnos que participan en las aulas y grupos de entrenamiento del simulador STACMO. Cada alumno puede ser asignado a un grupo dentro de un aula a través de la tabla GrupoMiembro.

Atributo	Descripción	Constraints
id_alumno	Identificador único del alumno.	PK
nombres	Nombres del alumno.	
apellidos	Apellidos del alumno.	
email	Dirección de correo electrónico institucional o personal.	UNIQUE
telefono	Número telefónico de contacto.	
fecha_nac	Fecha de nacimiento del alumno.	
DNI/CIP	Documento Nacional de Identidad o Carnet de Identificación Personal.	UNIQUE
genero	Género del alumno.	CHECK (M/F)
grado_militar	Grado militar o nivel académico del alumno, según corresponda.	CHECK (Cadete, Alférez, Teniente, etc.)

Relaciones:

1 Alumno — N GrupoMiembro (un alumno puede tener múltiples registros de participación, aunque en un aula solo puede pertenecer a un grupo).

Reglas

Cada alumno debe estar asignado a un único grupo dentro de la misma aula.
Los datos personales como email y DNI/CIP deben ser únicos en el sistema.

Tabla: EJERCICIO

Almacena la información de cada práctica o simulación realizada en el sistema STACMO. Está asociada a un Grupo y define las condiciones, escenario y objetivos bajo los cuales se desarrolla el entrenamiento. Además, registra los resultados y calificaciones obtenidas.

Atributo	Descripción	Constraints
id_ejercicio	Identificador único del ejercicio.	PK
fecha_inicio	Fecha y hora de inicio del ejercicio.	
fecha_fin	Fecha y hora de finalización del ejercicio.	
duracion_min	Duración total del ejercicio en minutos.	CHECK (duracion_min > 0)
tipo_ejercicio	Tipo o modalidad de ejercicio (ej. artillería, observación, morteros).	
comentarios	Observaciones adicionales sobre el desarrollo del ejercicio.	
nota	Calificación asignada al grupo en el ejercicio (escala 0–20).	CHECK (entre 0 AND 20)
id_grupo	Grupo que realizó el ejercicio.	FK
id_condiciones	Condiciones climáticas y ambientales bajo las cuales se realizó.	FK
id_condiciones	Condiciones climáticas y ambientales bajo las cuales se realizó.	FK

Relaciones:

1 Grupo — N Ejercicios (un grupo puede realizar múltiples ejercicios).
1 Ejercicio — 1 Condiciones (cada ejercicio ocurre bajo condiciones específicas).
1 Ejercicio — 1 Escenario (cada ejercicio se lleva a cabo en un escenario definido).
1 Ejercicio — N ObservacionesOA (registra las observaciones del OA).
1 Ejercicio — N ComandosCT (comandos de cálculo generados por el alumno).
1 Ejercicio — N ComandosSist (comandos de cálculo generados por el sistema).
1 Ejercicio — N StockMunicion (ejercicio puede consumir distintas municiones).
1 Ejercicio — N Objetivos (vía EjercicioObjetivo) (cada ejercicio puede tener varios objetivos asociados).

Reglas

Todo ejercicio debe estar asociado a un único grupo.
La calificación (nota) siempre debe estar en el rango 0–20.
Cada ejercicio debe estar vinculado obligatoriamente a un escenario y a unas condiciones.
Puede tener uno o varios objetivos, pero siempre debe haber al menos uno.

Tabla: OBJETIVO

Almacena la información de los objetivos militares o blancos usados en los ejercicios. Estos objetivos pueden ser tanques, posiciones enemigas, edificaciones u otras referencias geográficas. Se relacionan con los ejercicios a través de la tabla EjercicioObjetivo.

Atributo	Descripción	Constraints
id_objetivo	Identificador único del objetivo.	PK
tipo_objetivo	Tipo de objetivo (ejemplo: tanque, edificio, vehículo, coordenada).	CHECK (Tanques, Radar, Punto combustible, etc.)
coordenadas	Ubicación geográfica del objetivo en el escenario (puede ser UTM/lat-long).	
descripcion	Detalles adicionales o notas sobre el objetivo.	

Relaciones:

1 Objetivo — N Ejercicios (vía EjercicioObjetivo): un objetivo puede estar presente en múltiples ejercicios.
Se usa como entidad base para vincular blancos a las simulaciones.

Reglas

Cada objetivo debe tener tipo definido (no se admiten objetivos sin clasificar).
Las coordenadas deben registrarse siempre para poder ubicar el objetivo en el escenario.

Tabla: EJERCICIOOBJETIVO

Resuelve la relación muchos a muchos entre EJERCICIO y OBJETIVO. Cada registro vincula un objetivo específico a un ejercicio determinado. Permite que un ejercicio tenga varios objetivos y que un mismo objetivo sea reutilizado en muchos ejercicios.

Atributo	Descripción	Constraints
id_ejercicioobjetivo	Identificador único del vínculo ejercicio–objetivo.	PK
descripcion	Observaciones sobre el rol del objetivo dentro del ejercicio.	CHECK (Tanques, Radar, Punto combustible, etc.)
id_ejercicio	Ejercicio al que se asocia el objetivo.	FK
id_objetivo	Objetivo asociado al ejercicio.	FK

Relaciones:

1 Ejercicio — N EjercicioObjetivo (un ejercicio puede tener varios objetivos).
1 Objetivo — N EjercicioObjetivo (un objetivo puede ser usado en varios ejercicios).

Tabla: OBSERVACIÓNOA

Almacena las observaciones realizadas por el Observador Avanzado (OA) en cada ejercicio. Incluye rumbo, distancia y ángulo de situación para definir con precisión la ubicación del objetivo, junto con notas adicionales.

Atributo	Descripción	Constraints
id_observacion	Identificador único de la observación.	PK
rumbo	Dirección del objetivo expresada en mil rusos (0–5999). Se mide desde el norte en sentido horario.	CHECK (0 ≤ rumbo ≤ 5999)
distancia	Distancia al objetivo en metros desde la posición del Observador Avanzado	CHECK (distancia > 0)NULL
angulo_situacion	Ángulo vertical formado entre el plano horizontal del OA y la línea de visión dirigida hacia el objetivo.	CHECK (0 ≤ angulo_situacion ≤ 5999)
descripcion	Texto descriptivo o notas adicionales de la observación.	
tiempo	Marca de tiempo que indica cuanto demoró el proceso de observación.	
id_ejercicio	Identificador del ejercicio al que pertenece la observación.	FK

Relaciones:

1 Ejercicio — N ObservacionesOA
Cada ejercicio puede registrar múltiples observaciones realizadas por el Observador Adelantado (OA).
Cada observación pertenece exclusivamente a un único ejercicio.

Reglas

Cada observación debe estar asociada a un ejercicio válido (no puede existir una observación sin ejercicio).
Los valores de rumbo y ángulo de situación deben expresarse en milésimas rusas (0–5999) para mantener coherencia con el sistema de tiro ruso.
La distancia debe ser un valor positivo en metros (distancia > 0).
La marca de tiempo (tiempo) es obligatoria, pues cada observación debe estar cronológicamente registrada.
Un mismo ejercicio puede contener múltiples observaciones con distintos valores de rumbo, distancia y ángulo, pero todas deben compartir el id_ejercicio correspondiente.

Tabla: COMANDOCT

Registra el comando de tiro calculado por la Central de Tiro (CT) durante un ejercicio. Incluye los parámetros angulares y de elevación en mil rusos (0–5999), la marca temporal del cálculo y su asociación al ejercicio correspondiente.

Atributo	Descripción	Constraints
id_comando	Identificador único del comando de tiro calculado por la CT (alumno).	PK
deriva	Corrección/deflexión lateral del tiro en mil rusos.	CHECK (0 ≤ deriva ≤ 5999)
situacion	Dirección/rumbo del tubo en mil rusos.	CHECK (0 ≤ situacion ≤ 5999)
alza	Elevación del tubo en milesimos.	CHECK (alza ≥ 0)
marca_tiempo	Tiempo que demoro la central de tiro en hacer los cálculos.	
id_ejercicio	Identificador del ejercicio al que pertenece el comando calculado por la CT.	FK

Relaciones:

1 Ejercicio — N ComandoCT
Un ejercicio puede generar múltiples comandos calculados por la CT (alumnos)
Cada comando pertenece exclusivamente a un ejercicio.

Reglas

Todo ComandoCT debe pertenecer a un Ejercicio válido.
Unidades: deriva y situacion en mil rusos (0–5999); alza en mil (≥ 0).
marca_tiempo es obligatoria para poder medir la velocidad de los alumnos al hacerl los cálculos.

Tabla: COMANDOSIST

Almacena el comando de tiro matemáticamente correcto, calculado por el sistema mediante algoritmos programados. Este comando no es visible para los alumnos, solo por el instructor, quien lo utiliza como referencia para comparar con los comandos de tiro generados manualmente en la Central de Tiro (CT).

Atributo	Descripción	Constraints
id_comando_sist	Identificador único del comando generado por el sistema.	PK
deriva	Corrección/deflexión lateral del tiro en mil rusos.	CHECK (0 ≤ deriva ≤ 5999)
situacion	Dirección/rumbo del tubo en mil rusos.	CHECK (0 ≤ situacion ≤ 5999)
alza	Elevación del tubo en milesimos.	CHECK (alza ≥ 0)
id_ejercicio	Identificador del ejercicio al que pertenece el comando calculado por la CT.	FK

Relaciones:

1 Ejercicio — N ComandosSist
Un ejercicio puede generar múltiples comandos computados por el sistema.
Cada comando pertenece exclusivamente a un ejercicio.

Reglas

Cada comando generado debe estar asociado a un ejercicio válido.
Los valores de deriva, situación y alza se expresan en milésimas (0–5999), respetando el sistema de tiro ruso.
El comando del sistema es de uso exclusivo del instructor y no se comparte con los alumnos.
La función principal de este comando es servir como punto de comparación entre el cálculo correcto y los cálculos realizados por los alumnos de la CT.

Tabla: ESCENARIO

Almacena los distintos escenarios de entrenamiento en los que se desarrollan los ejercicios. Cada escenario define un entorno específico (ejemplo: costa, sierra, selva) que condiciona las prácticas y permite evaluar a los grupos en contextos variados.

Atributo	Descripción	Constraints
id_escenario	Identificador único del escenario.	PK
nombre_esc	Nombre único y descriptivo del escenario	UNIQUE

Relaciones:

1 Escenario — N Ejercicios
Cada escenario puede estar asociado a múltiples ejercicios.
Cada ejercicio se desarrolla en un único escenario específico.

Reglas

El nombre del escenario debe ser único para evitar duplicados y facilitar la clasificación.
Todo ejercicio debe estar asociado obligatoriamente a un escenario válido (id_escenario no puede ser nulo).
Los escenarios representan categorías predefinidas y sirven como referencia para el análisis de desempeño en distintos contextos de entrenamiento.

Tabla: CONDICIONES

Almacena las condiciones ambientales bajo las cuales se ejecuta un ejercicio en el simulador: viento, lluvia, niebla y hora del día. Funciona como catálogo reutilizable para que varios ejercicios puedan referirse a la misma combinación de condiciones.

Atributo	Descripción	Constraints
id_condiciones	Identificador único del registro de condiciones.	PK
viento	Intensidad del viento durante el ejercicio.	CHECK (NINGUNA,BAJA,-MEDIA,ALTA)
lluvia	Nivel de precipitación presente en el ejercicio.	CHECK (NINGUNA,BAJA,-MEDIA,ALTA)
niebla	Densidad de la niebla en el ejercicio, afecta la visión del OA.	CHECK (NINGUNA,BAJA,-MEDIA,ALTA)
hora_del_dia	Momento del día en que se ejecuta el ejercicio en el mundo virtual.	CHECK (DIA, TARDE, NOCHE)

Relaciones:

1 Condiciones — N Ejercicios
Un registro de Condiciones puede ser utilizado por muchos ejercicios.

Reglas

Las cuatro dimensiones (viento, lluvia, niebla, hora_del_dia) deben estar siempre definidas.

Tabla: MUNICION

Almacena la información general sobre los diferentes tipos de munición disponibles en el simulador. Define sus características principales (tipo, calibre y descripción), que luego se usan para gestionar el StockMunicion y asociar consumos en los ejercicios.

Atributo	Descripción	Constraints
id_municion	Identificador único de la munición.	PK
tipo_municion	Clasificación del tipo de munición.	CHECK (Explosiva, Fumigena)
calibre	Calibre de la granada(en mm).	CHECK (120, 122)
descripcion	Información adicional sobre el uso o características de la munición.	

Relaciones:

1 Municion — N StockMunicion
Una munición puede estar registrada en múltiples lotes de stock disponibles.
Cada registro en StockMunicion hace referencia a una única munición

Reglas

Cada tipo de munición debe estar previamente registrado en esta tabla antes de poder gestionarse en el stock.

Tabla: STOCKMUNICION

Almacena los registros de disponibilidad y consumo de cada tipo de munición dentro de un ejercicio específico. Permite llevar el control de las cantidades iniciales, usadas y finales, vinculadas tanto al tipo de munición como al ejercicio en que se emplea.

Atributo	Descripción	Constraints
id_stock	Identificador único del registro de stock de munición.	PK
cant_inicial	Cantidad inicial de munición disponible al inicio del	CHECK (cant_inicial ≥ 0)
cant_usada	Cantidad de munición consumida durante el ejercicio.	CHECK (cant_usada ≥ 0)
cant_final	Cantidad restante de munición al finalizar el ejercicio.	CHECK (cant_final = cant_inicial - cant_usada cant_final ≥ 0)
id_ejercicio	Identificador del ejercicio asociado al consumo de la munición.	FK
id_municion	Identificador del tipo de munición al que corresponde el registro de stock.	FK

Relaciones:

1 Ejercicio — N StockMunicion
Cada ejercicio puede registrar múltiples consumos de distintos tipos de munición.
1 Municion — N StockMunicion
Cada tipo de munición puede estar presente en diferentes ejercicios o lotes de stock.

Reglas

cant_final debe calcularse automáticamente como cant_inicial - cant_usada.
No se permite registrar un stock con cantidades negativas.

Conclusiones, Lecciones Aprendidas

El desarrollo del modelo lógico del sistema STACMO ha marcado un avance decisivo en la construcción de la base de datos que sustenta el simulador. A diferencia del modelo conceptual, donde se trabajó con abstracciones, esta etapa permitió definir de manera precisa las tablas que representan cada entidad, junto con sus atributos, claves primarias y foráneas, y las restricciones necesarias para garantizar la integridad de los datos.

Este proceso dio como resultado un diagrama mucho más cercano a la implementación real, en el que cada relación y cardinalidad quedó claramente especificada. La definición de constraints, no solo fortaleció la solidez técnica del modelo, sino que además reflejó de forma fiel las reglas de negocio que guían el funcionamiento del simulador. Gracias a ello, se asegura que la información almacenada sea consistente, válida y trazable, respondiendo a las exigencias de un entorno de entrenamiento militar.

El modelo lógico, además, permitió comprobar que todos los actores del simulador el Observador Avanzado, la Central de Tiro, los Instructores, las Piezas Virtuales, los objetivos y las condiciones del entorn, se encuentran correctamente representados y vinculados. De esta manera, se logró estructurar un ecosistema de datos que reproduce con fidelidad las dinámicas del entrenamiento en artillería de campaña y morteros.

El paso del modelo conceptual al lógico evidenció la necesidad de trabajar con un enfoque iterativo y de validación constante. Fue indispensable revisar y ajustar las relaciones entre entidades, las claves foráneas y las restricciones para asegurar que todo el sistema mantuviera coherencia interna. También quedó claro que la documentación detallada como las fichas resumen elaboradas para cada tabla, es un recurso valioso, ya que facilita la comprensión del modelo y garantiza que no se pierdan de vista los detalles más importantes.

Otra lección significativa fue comprender que las restricciones no son simples reglas técnicas, sino mecanismos que traducen al lenguaje de la base de datos las normas de negocio y las prácticas propias del entrenamiento militar. Cada constraint incorporado asegura que la base de datos actúe como un guardián de la disciplina y la precisión que exige el simulador.

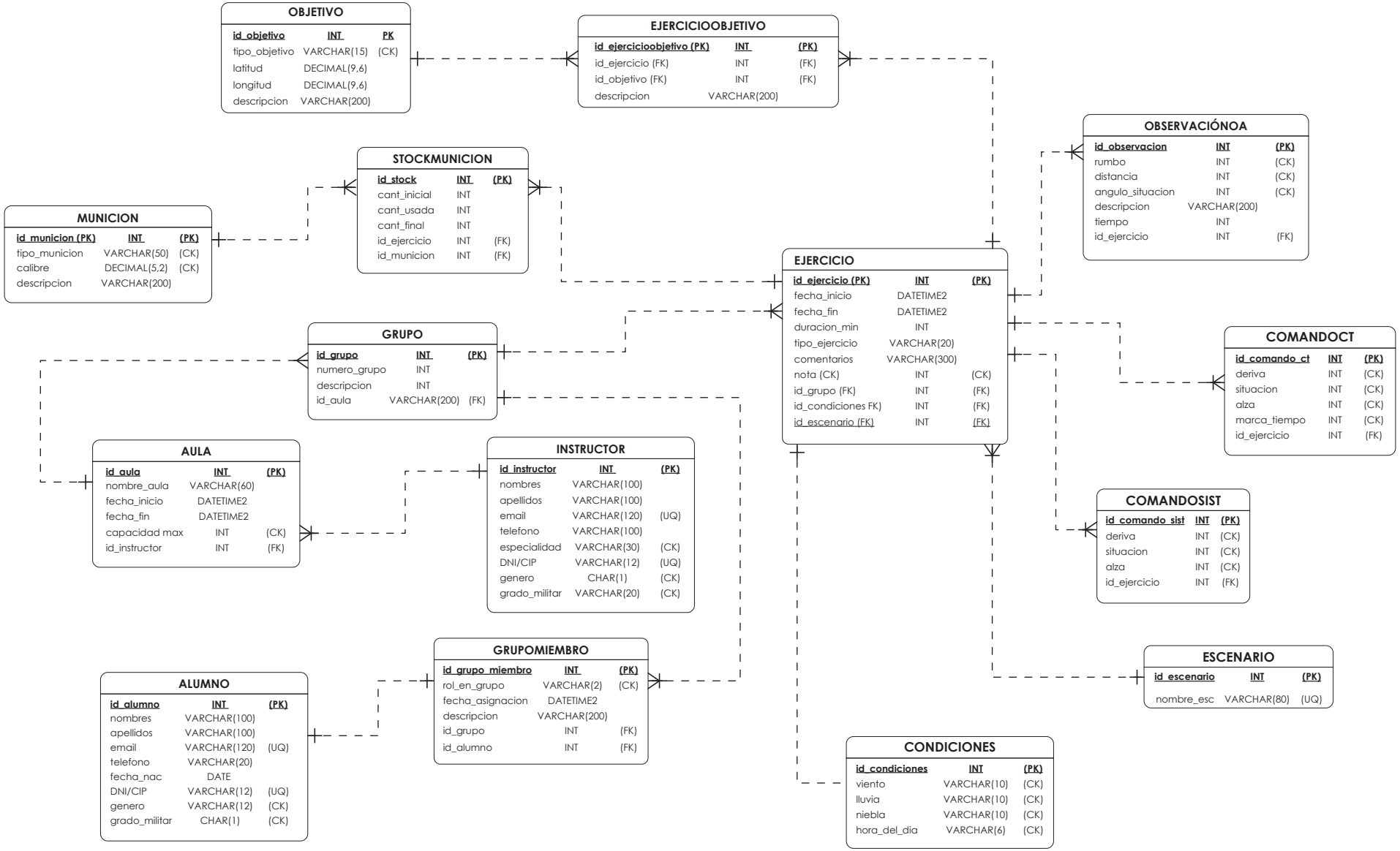
Cierre de la Primera Entrega

Con la finalización de esta primera entrega, se consolida una base firme sobre la cual avanzar hacia la siguiente etapa del proyecto. Haber definido el modelo lógico significa haber construido la columna vertebral del sistema de datos de STACMO, un soporte que no solo es robusto y coherente, sino también escalable y adaptable a futuras necesidades.

Este trabajo demuestra cómo es posible trasladar la complejidad operativa de un simulador militar a un diseño estructurado de base de datos, donde cada interacción entre alumnos, instructores y sistema puede ser registrada, analizada y evaluada con precisión. Con ello, no solo se cumple con los objetivos académicos planteados, sino que también se fortalece la capacidad del simulador como herramienta de modernización y autonomía tecnológica en el entrenamiento del Ejército del Perú.

La entrega cierra así con un resultado tangible: un modelo lógico sólido, validado y documentado, que prepara el terreno para el siguiente gran paso, la creación del modelo físico y su implementación práctica.

Diagrama Físico



Explicación del Diagrama Físico

Tabla: INSTRUCTOR

Contiene los datos de los instructores que dirigen y supervisan los ejercicios dentro del simulador STACMO. Cada aula está obligatoriamente asignada a un instructor.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_instructor (PK)	INT	INT	PK, IDENTITY(1,1)
nombres	VARCHAR(100)	NO	
apellidos	VARCHAR(100)	NO	
email (UQ)	VARCHAR(120)	NO	UNIQUE
telefono	VARCHAR(100)	SI	
especialidad (CK)	VARCHAR(30)	NO	CHECK (ARTILLERíA, INFANTERíA, ETC)
DNI/CIP (UQ)	VARCHAR(12)	SI	UNIQUE (almacena DNI o CIP; UNICO
genero (CK)	CHAR(1)	NO	CHECK (M/F)
grado_militar (CK)	VARCHAR(20)	NO	CHECK (Cadete, Alférez, Teniente, etc.)

Tabla: AULA

Representa el espacio académico en el que se organiza un conjunto de alumnos bajo la dirección de un instructor. Cada aula está compuesta por grupos de trabajo y constituye la unidad básica de organización para realizar los ejercicios en el simulador STACMO.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_aula	INT	NO	PK, IDENTITY(1,1)
id_instructor	INT	NO	FK → INSTRUCTOR(id_instructor)
nombre_aula	VARCHAR(60)	NO	UNIQUE (no se repite el nombre)
fecha_inicio	DATETIME2	NO	CHECK: echa_fin >= fecha_inicio
fecha_fin	DATETIME2	SI	CHECK BETWEEN 4 AND 60
capacidad_max	INT	NO	CHECK (capacidad_max > 0)

Tabla: GRUPO

Subconjunto de alumnos dentro de un Aula con el que se ejecutan los Ejercicios. Cada grupo se identifica por un número que es único dentro de su aula (p. ej., 1, 2, 3). Un grupo está compuesto por 4 miembros (1 Observador Avanzado (OA) + 3 Miembros de la Central de Tiro (CT).

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_grupo	INT	NO	PK, IDENTITY(1,1)
id_aula	INT	NO	FK
numero_grupo	INT	NO	CHECK BETWEEN 1 AND 99
descripcion	VARCHAR(200)	SI	

Tabla: GRUPOMIEMBRO

Registra a los alumnos que integran un grupo dentro de un aula y el rol que cumplen en ese grupo (OA o CT). Es la tabla asociativa que materializa la relación Alumno–Grupo.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_grupo_miembro	INT	NO	PK, IDENTITY(1,1)
id_grupo	INT	NO	FK
id_alumno	INT	NO	FK
rol_en_grupo	VARCHAR(2)	NO	CHECK (OA, CT)
fecha_asignacion	DATETIME2	NO	DEFAULT
descripcion	VARCHAR(200)	SI	

Tabla: ALUMNO

Almacena la información personal y académica de los alumnos que participan en las aulas y grupos de entrenamiento del simulador STACMO. Cada alumno puede ser asignado a un grupo dentro de un aula a través de la tabla GrupoMiembro.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_alumno	INT	NO	PK, IDENTITY(1,1)
nombres	VARCHAR(100)	NO	
apellidos	VARCHAR(100)	NO	
email	VARCHAR(120)	NO	UNIQUE
telefono	VARCHAR(20)	SI	
fecha_nac	DATE	NO	
DNI/CIP	VARCHAR(12)	SI	UNIQUE
genero	CHAR(1)	NO	CHECK (M/F)
grado_militar	VARCHAR(20)	NO	CHECK (Cadete, Alférez, Teniente, etc.)

Tabla: EJERCICIO

Almacena la información de cada práctica o simulación realizada en el sistema STACMO. Está asociada a un Grupo y define las condiciones, escenario y objetivos bajo los cuales se desarrolla el entrenamiento. Además, registra los resultados y calificaciones obtenidas.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_ejercicio	INT	NO	PK, IDENTITY(1,1)
fecha_inicio	DATETIME2	NO	
fecha_fin	DATETIME2	SI	
duracion_min	INT (calc.)	SI	CHECK (duracion_min > 0)
tipo_ejercicio	VARCHAR(20)	NO	
comentarios	VARCHAR(300)	SI	
nota	INT	SI	CHECK (entre 0 AND 20)
id_grupo	INT	NO	FK
id_condiciones	INT	NO	FK
id_condiciones	INT	NO	FK

Tabla: OBJETIVO

Almacena la información de los objetivos militares o blancos usados en los ejercicios. Estos objetivos pueden ser tanques, posiciones enemigas, edificaciones u otras referencias geográficas. Se relacionan con los ejercicios a través de la tabla EjercicioObjetivo.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_objetivo	INT	NO	PK, IDENTITY(1,1)
tipo_objetivo	VARCHAR(15)	NO	CHECK (Tanques, Radar, Punto combustible, etc.)
latitud	DECIMAL(9,6)	NO	CHECK
longitud	DECIMAL(9,6)	NO	CHECK
descripcion	VARCHAR(200)	SI	

Tabla: EJERCICIOOBJETIVO

Resuelve la relación muchos a muchos entre EJERCICIO y OBJETIVO. Cada registro vincula un objetivo específico a un ejercicio determinado. Permite que un ejercicio tenga varios objetivos y que un mismo objetivo sea reutilizado en muchos ejercicios.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_ejercicioobjetivo	INT	NO	PK
descripcion	VARCHAR(200)	SI	CHECK (Tanques, Radar, Punto combustible, etc.)
id_ejercicio	INT	NO	FK
id_objetivo	INT	NO	FK

Tabla: OBSERVACIÓNOA

Almacena las observaciones realizadas por el Observador Avanzado (OA) en cada ejercicio. Incluye rumbo, distancia y ángulo de situación para definir con precisión la ubicación del objetivo, junto con notas adicionales.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_observacion	INT	NO	PK, IDENTITY(1,1)
rumbo	INT	NO	CHECK (0 ≤ rumbo ≤ 5999)
distancia	INT	NO	CHECK (distancia > 0)NULL
angulo_situacion	INT	NO	CHECK (0 ≤ angulo_situacion ≤ 5999)
descripcion	VARCHAR(200)	SI	
tiempo	Marca de tiempo que	NO	CHECK >= 0
id_ejercicio	INT	NO	FK

Tabla: COMANDOCT

Registra el comando de tiro calculado por la Central de Tiro (CT) durante un ejercicio. Incluye los parámetros angulares y de elevación en mil rusos (0–5999), la marca temporal del cálculo y su asociación al ejercicio correspondiente.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_comando	INT	NO	PK
deriva	INT	NO	CHECK (0 ≤ deriva ≤ 5999)
situacion	INT	NO	CHECK (0 ≤ situacion ≤ 5999)
alza	INT	NO	CHECK (alza ≥ 0)
marca_tiempo	INT	NO	
id_ejercicio	INT	NO	FK

Tabla: COMANDOSIST

Almacena el comando de tiro matemáticamente correcto, calculado por el sistema mediante algoritmos programados. Este comando no es visible para los alumnos, solo por el instructor, quien lo utiliza como referencia para comparar con los comandos de tiro generados manualmente en la Central de Tiro (CT).

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_comando_sist	INT	NO	PK
deriva	INT	NO	CHECK (0 ≤ deriva ≤ 5999)
situacion	INT	NO	CHECK (0 ≤ situacion ≤ 5999)
alza	INT	NO	CHECK (alza ≥ 0)
id_ejercicio	INT	NO	FK

Tabla: ESCENARIO

Almacena los distintos escenarios de entrenamiento en los que se desarrollan los ejercicios. Cada escenario define un entorno específico (ejemplo: costa, sierra, selva) que condiciona las prácticas y permite evaluar a los grupos en contextos variados.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_escenario	INT	NO	PK
nombre_esc	VARCHAR(80)	NO	UNIQUE

Tabla: CONDICIONES

Almacena las condiciones ambientales bajo las cuales se ejecuta un ejercicio en el simulador: viento, lluvia, niebla y hora del día. Funciona como catálogo reutilizable para que varios ejercicios puedan referirse a la misma combinación de condiciones.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_condiciones	INT	NO	PK
viento	VARCHAR(10)	NO	CHECK (NINGUNA,BAJA,MEDIA,ALTA)
lluvia	VARCHAR(10)	NO	CHECK (NINGUNA,BAJA,MEDIA,ALTA)
niebla	VARCHAR(10)	NO	CHECK (NINGUNA,BAJA,MEDIA,ALTA)
hora_del_dia	VARCHAR(6)	NO	CHECK (DIA, TARDE, NOCHE)

Tabla: MUNICION

Almacena la información general sobre los diferentes tipos de munición disponibles en el simulador. Define sus características principales (tipo, calibre y descripción), que luego se usan para gestionar el StockMunicion y asociar consumos en los ejercicios.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_municion	INT	NO	PK
tipo_municion	VARCHAR(50)	NO	CHECK (Explosiva, Fumigena)
calibre	DECIMAL(5,2)	NO	CHECK (120, 122)
descripcion	VARCHAR(200)	SI	

Tabla: STOCKMUNICION

Almacena los registros de disponibilidad y consumo de cada tipo de munición dentro de un ejercicio específico. Permite llevar el control de las cantidades iniciales, usadas y finales, vinculadas tanto al tipo de munición como al ejercicio en que se emplea.

Atributo	Tipo de dato	Nulable	Constraints/ Comentarios
id_stock	INT	NO	PK
cant_inicial	INT	NO	CHECK (cant_inicial ≥ 0)
cant_usada	INT	NO	CHECK (cant_usada ≥ 0)
cant_final	INT	SI	CHECK (cant_final = cant_inicial - cant_usada cant_final ≥ 0)
id_ejercicio	INT	NO	FK
id_municion	INT	NO	FK

Consultas y Optimización

1. Contexto y propósito de esta sección

Como continuación del trabajo de análisis e implementación (Avances 1 y 2), en esta fase del proyecto se buscó evidenciar de manera práctica que la base de datos STACMODEMO no solo está correctamente desplegada, sino que también es capaz de operar, generar reportes útiles y mantener integridad cuando se ejecutan consultas y operaciones típicas de un entorno real. Esta parte de la entrega se centra en demostrar cinco evidencias (A–E) asociadas a operación y programación: generación de reportes a través de una vista, uso de subconsultas para métricas en línea, aplicación de índices y validación mediante planes de ejecución para sustentar optimización, y finalmente el uso de procedimientos almacenados con transacciones (explícitas e implícitas) para asegurar consistencia ante escenarios válidos e inválidos.

El objetivo no fue "hacer consultas sueltas", sino presentar un conjunto coherente de evidencias donde se vea claramente que la base tiene un uso orientado a decisión: consolidación de información, medición de desempeño (nota), consumo operativo (munición), y control transaccional para evitar estados inconsistentes. Todo lo mostrado se sustenta ejecutando scripts en SQL Server (SSMS) y validando resultados en pantalla (resultsets y Execution Plan).

A) VISTA — Reporte de ejercicios con instructor, escenario, nota y consumo

En este punto se evidencia la capacidad de STACMODEMO para entregar un reporte consolidado de manera directa, legible y reutilizable, sin necesidad de reescribir joins complejos cada vez que se quiera consultar la información. La vista se plantea como un "reporte guardado" a nivel de base de datos: una vez definida, cualquier consulta futura puede consumirla como si fuera una tabla lógica. Esto es importante porque, en un contexto real, los reportes se consultan recurrentemente y deben estar estandarizados, especialmente cuando involucran múltiples tablas del modelo.

La vista VM_REPORTE_EJERCICIOS integra la tabla central EJERCICIO (sesiones), la estructura organizativa GRUPO y AULA, el responsable INSTRUCTOR y el componente contextual ESCENARIO. Con esta consolidación, se obtiene un panorama de "qué ejercicio fue, cuándo ocurrió, quién lo condujo, dónde se realizó, bajo qué escenario y con qué nota". Además, el reporte incorpora una métrica operativa clave: total de munición usada por ejercicio, calculada dentro de la vista mediante una subconsulta correlacionada sobre STOCKMUNICION. Así, el reporte no solo presenta datos descriptivos, sino también una medición cuantitativa asociada al consumo.

En términos de sustento, este punto se valida al ejecutar la creación/alteración de la vista y luego hacer un SELECT a la vista, mostrando el resultado como tabla de reporte. La idea es que la profesora vea que el reporte está correctamente armado y que efectivamente sale desde el modelo, no "inventado".

Script A (creación de vista + consulta del reporte)

```
USE STACMODEMO;
GO

-- 1) VISTA (reporte)
CREATE OR ALTER VIEW dbo.VM_REPORTE_EJERCICIOS
AS
SELECT
    e.id_ejercicio,
    e.fecha_inicio,
    e.tipo_ejercicio,
    e.nota,
    g.id_grupo,
    a.nombre_aula,
    i.nombres + ' ' + i.apellidos AS instructor,
    es.nombre_esc AS escenario,
    -- Consumo total (subconsulta correlacionada)
    (SELECT ISNULL(SUM(sm.cant_usada),0)
     FROM dbo.STOCKMUNICION sm
     WHERE sm.id_ejercicio = e.id_ejercicio) AS total_municion_usada
FROM dbo.EJERCICIO e
JOIN dbo.GRUPO g      ON g.id_grupo = e.id_grupo
JOIN dbo.AULA a       ON a.id_aula = g.id_aula
JOIN dbo.INSTRUCTOR i ON i.id_instructor = a.id_instructor
JOIN dbo.ESCENARIO es ON es.id_escenario = e.id_escenario;
GO

-- Mostrar el reporte
SELECT TOP (20) *
FROM dbo.VM_REPORTE_EJERCICIOS
ORDER BY fecha_inicio;
GO
```

B) SUBCONSULTA — Top 3 ejercicios con mejor nota y su munición

En este punto se evidencia un recurso muy útil para reportes reales: la posibilidad de calcular un valor adicional "en línea" dentro del SELECT, sin depender de una columna almacenada o de una tabla derivada adicional. La subconsulta correlacionada permite, para cada ejercicio listado, calcular automáticamente el consumo de munición asociado en STOCKMUNICION. Esta lógica es especialmente valiosa cuando se quiere construir un reporte rápido que mezcle "indicadores de desempeño" con "uso de recursos", sin complicar demasiado el query principal.

El reporte "Top 3 por nota" cumple un objetivo claro de análisis: identificar las sesiones mejor calificadas y asociarlas con el consumo. Desde una perspectiva de toma de decisiones, este tipo de consulta podría ayudar a responder preguntas como: ¿qué tipo de ejercicios tienden a tener mejor desempeño? ¿hay relación entre consumo y resultado? Aunque en esta entrega se presenta en modo demostración, el valor está en evidenciar que el modelo soporta consultas orientadas a análisis.

La evidencia se sustenta ejecutando el query y mostrando el resultset con 3 filas, donde se ve explícitamente la columna calculada municion_usada. Aquí el énfasis no está en la cantidad de datos, sino en demostrar la técnica y su utilidad.

Script B (subconsulta Top 3)

```
USE STACMODEMO;
GO

-- 2) SUBCONSULTA (Top ejercicios por nota + consumo)
SELECT TOP (3)
    e.id_ejercicio,
    e.tipo_ejercicio,
    e.nota,
    (SELECT ISNULL(SUM(sm.cant_usada),0)
     FROM dbo.STOCKMUNICION sm
     WHERE sm.id_ejercicio = e.id_ejercicio) AS municion_usada
FROM dbo.EJERCICIO e
ORDER BY e.nota DESC, e.id_ejercicio;
GO
```

C) ÍNDICE + PLAN DE EJECUCIÓN — Evidencia de optimización para reportes

En este punto se evidencia que la base no solo consulta datos, sino que también puede optimizarse para patrones de consulta típicos. En un entorno real, los reportes suelen filtrar por criterios repetidos (por ejemplo, grupo, fechas, rangos) y ordenar por columnas específicas (por ejemplo, fecha). Si el motor no cuenta con un índice alineado a ese patrón, normalmente tendrá que leer más datos, ordenar con mayor trabajo y, en bases grandes, generar costos innecesarios.

La consulta utilizada representa un escenario muy realista: listar ejercicios de un grupo específico (id_grupo = 1) y devolverlos en orden cronológico (ORDER BY fecha_inicio). La evidencia se basa en comparar el plan de ejecución real antes y después de crear un índice pensado exactamente para ese patrón: (id_grupo, fecha_inicio).

En tu evidencia visual se observa un cambio totalmente válido y claro:

Antes del índice: el plan muestra un Clustered Index Scan y un operador Sort. Esto suele ocurrir cuando el motor no tiene un índice que le permita filtrar y devolver ya ordenado; por ello lee más ampliamente y luego ordena.

Después del índice: aparece Index Seek (NonClustered) usando el índice creado. Esto significa que el motor ya no "recorre" de forma amplia, sino que puede ir directo al rango de filas del grupo solicitado. Además, como el índice está compuesto por id_grupo y luego fecha_inicio, queda mejor preparado para devolver resultados ya estructurados para el ordenamiento.

En tu caso también aparece un Key Lookup (Clustered), y esto no es un error: ocurre cuando el índice no contiene todas las columnas requeridas por el SELECT, entonces el motor usa el índice para ubicar rápido y luego "consulta" la tabla base (clustered/PK) para recuperar columnas faltantes. Para efectos de evidencia académica, este comportamiento es totalmente aceptable porque lo que se requiere demostrar es que el motor sí usa el índice y que el plan refleja un acceso más dirigido (Seek), confirmando optimización para el patrón consultado.

Lo más importante aquí es que la evidencia se sustenta de forma visual en SSMS, mostrando la pestaña Execution Plan para "ANTES" y para "DESPUÉS", y señalando la diferencia en los operadores principales.

Script C (antes y después + control de existencia del índice)

```
USE STACMODEMO;
GO

-- 0) (Opcional) Ver si existe el índice
SELECT i.name AS indice
FROM sys.indexes i
WHERE i.object_id = OBJECT_ID('dbo.EJERCICIO')
  AND i.name = 'IDX_EJERCICIO_id_grupo_fecha_inicio';
GO

-- 1) Borrar el índice si ya existe
IF EXISTS (
    SELECT 1
    FROM sys.indexes
    WHERE object_id = OBJECT_ID('dbo.EJERCICIO')
      AND name = 'IDX_EJERCICIO_id_grupo_fecha_inicio'
)
BEGIN
    DROP INDEX IDX_EJERCICIO_id_grupo_fecha_inicio ON dbo.EJERCICIO;
END
GO

-- 2) CONSULTA ANTES (con Ctrl+M activado)
PRINT 'ANTES DEL INDICE';
SELECT e.id_ejercicio, e.id_grupo, e.fecha_inicio, e.nota
FROM dbo.EJERCICIO e
WHERE e.id_grupo = 1
ORDER BY e.fecha_inicio;
GO

-- 3) Crear índice
PRINT 'CREANDO INDICE';
CREATE INDEX IDX_EJERCICIO_id_grupo_fecha_inicio
ON dbo.EJERCICIO (id_grupo, fecha_inicio);
GO

-- 4) CONSULTA DESPUÉS (con Ctrl+M activado)
PRINT 'DESPUES DEL INDICE';
SELECT e.id_ejercicio, e.id_grupo, e.fecha_inicio, e.nota
FROM dbo.EJERCICIO e
WHERE e.id_grupo = 1
ORDER BY e.fecha_inicio;
GO

-- 5) Evidencia extra: listar índices
EXEC sp_helpindex 'dbo.EJERCICIO';
GO
```

D) PROCEDIMIENTO + TRANSACCIÓN EXPLÍCITA — Registrar consumo de munición con control de integridad

Este punto evidencia un aspecto clave de "operación y programación" de base de datos: no basta con insertar datos; se requiere asegurar que las operaciones mantengan consistencia, especialmente cuando hay reglas críticas del negocio. En este caso, la regla fundamental es que el stock final (cant_final) no puede quedar negativo, porque eso implicaría un registro inconsistente (se consumió más de lo disponible). Para sostener esa consistencia, se implementa un procedimiento almacenado que utiliza transacción explícita (BEGIN TRAN / COMMIT / ROLLBACK) y un bloque TRY/CATCH para controlar tanto escenarios válidos como errores.

La estructura del procedimiento está diseñada para evidenciar buenas prácticas:

Se inicia la transacción con BEGIN TRAN.

Se calcula cant_final = cant_inicial - cant_usada.

Si el resultado es negativo, se genera un error con RAISERROR.

Si es válido, se hace el INSERT en STOCKMUNICION.

Se confirma con COMMIT.

Si ocurre cualquier error, se ejecuta ROLLBACK, evitando que se inserte información inconsistente, y se devuelve un mensaje de error.

La evidencia se presenta con dos ejecuciones:

Un caso correcto (OK) donde se inserta y se confirma.

Un caso incorrecto donde el consumo dejaría el stock negativo, por lo tanto se dispara el error y la transacción revierte (rollback). El sustento visual está en el mensaje devuelto por el procedimiento y en verificar que el registro inválido no aparece al consultar la tabla (por ejemplo, con un TOP ordenado por id_stock DESC).

Script D (SP + caso OK + validación + caso ERROR)

```
USE STACMODEMO;
GO

-- 4) PROCEDIMIENTO con TRANSACCIÓN EXPLÍCITA
CREATE OR ALTER PROCEDURE dbo.SP_REGISTRAR_CONSUMO_MUNICION
   @id_ejercicio INT,
   @id_municion  INT,
   @cant_inicial INT,
   @cant_usada   INT
AS
BEGIN
   SET NOCOUNT ON;

   BEGIN TRY
      BEGIN TRAN;

      DECLARE @cant_final INT = @cant_inicial - @cant_usada;

      IF (@cant_final < 0)
      BEGIN
         RAISERROR('Error: cant_final no puede ser negativo.', 16, 1);
      END

      INSERT INTO dbo.STOCKMUNICION (cant_inicial, cant_usada, cant_final, id_ejercicio, id_municion)
      VALUES (@cant_inicial, @cant_usada, @cant_final, @id_ejercicio, @id_municion);

      COMMIT TRAN;

      SELECT 'OK - Consumo registrado' AS resultado, @cant_final AS cant_final;
   END TRY
   BEGIN CATCH
      IF @@TRANCOUNT > 0 ROLLBACK TRAN;

      SELECT
         'ERROR - Se hizo ROLLBACK' AS resultado,
         ERROR_NUMBER() AS error_numero,
         ERROR_MESSAGE() AS detalle_error;
   END CATCH
END
GO

-- Caso OK
EXEC dbo.SP_REGISTRAR_CONSUMO_MUNICION
   @id_ejercicio = 1,
   @id_municion  = 1,
   @cant_inicial = 50,
   @cant_usada   = 10;
GO

-- Evidencia: ver el último registro insertado
SELECT TOP (5) *
FROM dbo.STOCKMUNICION
ORDER BY id_stock DESC;
GO

-- Caso ERROR (debe fallar y hacer rollback)
EXEC dbo.SP_REGISTRAR_CONSUMO_MUNICION
   @id_ejercicio = 1,
   @id_municion  = 1,
   @cant_inicial = 5,
   @cant_usada   = 20; -- negativo => error
GO
```

E) TRANSACCIÓN IMPLÍCITA — Demostración breve sin complejidad

Este punto se presenta como demostración complementaria para cubrir el requerimiento de "transacciones implícitas" de forma simple. A diferencia de la transacción explícita (donde el programador inicia y cierra la transacción), en modo implícito SQL Server inicia automáticamente una transacción cuando se ejecuta una operación de modificación (INSERT/UPDATE/DELETE). El usuario debe cerrarla manualmente con COMMIT o ROLLBACK.

La evidencia aquí es directa: se activa SET IMPLICIT_TRANSACTIONS ON, se ejecuta un INSERT de prueba (en este caso en OBJETIVO) y luego se cierra con COMMIT. Finalmente se desactiva el modo para no afectar otras ejecuciones. Este bloque se muestra como un respaldo de conocimiento del comportamien-

Script E (implícita ON + INSERT + COMMIT + OFF)

```
USE STACMODEMO;
GO

-- Transacción implícita (demostración)
SET IMPLICIT_TRANSACTIONS ON;
GO

INSERT INTO dbo.OBJETIVO (tipo_objetivo, latitud, longitud, descripcion)
VALUES ('OTRO', -12.050000, -77.050000, 'Objetivo demo transacción implícita');

COMMIT;
GO

SET IMPLICIT_TRANSACTIONS OFF;
GO
```

Respaldo y restauración de la base de datos STACMODEMO (SQL Server)

En esta etapa se implementa y evidencia un plan de respaldo completo (FULL BACKUP) y su restauración desde un archivo .BAK, con el objetivo de asegurar la continuidad operativa del sistema y la capacidad de recuperación ante incidentes. El respaldo completo consolida la estructura y los datos existentes en la base de datos al momento de la ejecución. Posteriormente, la restauración valida que el contenido respaldado puede reconstruirse de forma consistente en el servidor, garantizando integridad y disponibilidad.

La demostración considera tres momentos: (1) identificación de una ruta válida para la generación del

1. Determinación de la ruta de respaldo del servidor

Como parte del procedimiento, se define una ruta de respaldo válida en el servidor para asegurar que el archivo .bak se genere en un directorio existente y accesible para el servicio de SQL Server. En este caso, se creó una carpeta dedicada en el disco local:

Ruta utilizada: C:\Backup\

De esta manera, el respaldo queda centralizado y es fácil de ubicar y reutilizar en el proceso de restauración, manteniendo la evidencia ordenada dentro de un directorio específico.

2. Respaldo completo (FULL BACKUP) de STACMODEMO

Se ejecuta un respaldo completo que genera el archivo STACMODEMO_FULL.bak. Se utiliza INIT para asegurar un archivo limpio por ejecución y STATS para evidenciar el progreso. Adicionalmente, se aplica CHECKSUM como validación interna del respaldo, y COMPRESSION cuando está disponible en la instan-

Script (backup full):

```
USE master;
GO

BACKUP DATABASE STACMODEMO
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\STACMODEMO_FULL.bak'
WITH
   INIT,
   COMPRESSION,
   CHECKSUM,
   STATS = 10;
GO
```

3. Verificación del archivo de respaldo (.BAK)

Para sustentar que el archivo de respaldo es consistente y puede ser restaurado, se realiza una verificación del backup mediante RESTORE VERIFYONLY. Esta validación permite evidenciar que el archivo .bak es legible y apto para restauración.

Script (verificación):

```
USE master;
GO

RESTORE VERIFYONLY
FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\STACMODEMO_FULL.-
bak';
GO
```

4. Restauración controlada del backup en una base alterna

Para demostrar la restauración sin afectar la base operativa, el respaldo se restaura como una base alterna denominada STACMODEMO_RESTORE. Previamente, se identifican los nombres lógicos de los archivos contenidos en el backup con RESTORE FILELISTONLY, ya que serán utilizados en la instrucción MOVE para reconstruir el .mdf y el .ldf en rutas del servidor.

Script (identificar nombres lógicos):

```
USE master;
GO

RESTORE FILELISTONLY
FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\STACMODEMO_FULL.-
bak';
GO
```

Luego, se restaura la base como STACMODEMO_RESTORE, moviendo los archivos de datos y log. En el caso de que la base alterna ya exista, se elimina para asegurar una restauración limpia y trazable.

5. Validación posterior a la restauración

Finalmente, se sustenta la restauración verificando que la base restaurada contiene estructura y datos. Para ello se consultan: (a) el nombre de la base activa, (b) el total de tablas creadas, y (c) el total de registros por tabla. Esto evidencia que el respaldo no solo creó la base sino que recuperó tablas y datos de manera consistente.

Script (validación post-restore):

```
USE STACMODEMO_RESTORE;
GO

SELECT DB_NAME() AS bd_actual;
SELECT COUNT(*) AS total_tablas FROM sys.tables;

SELECT t.name AS tabla, SUM(p.rows) AS total_registros
FROM sys.tables t
JOIN sys.partitions p ON p.object_id = t.object_id
WHERE p.index_id IN (0,1)
GROUP BY t.name
ORDER BY t.name;
GO
```

Conclusiones generales del proyecto del ciclo

Este ciclo confirmó, con evidencias concretas, que una base de datos bien diseñada no es solo un requisito académico: es el componente que permite que un simulador como STACMO convierta el entrenamiento en información medible, trazable y útil para la toma de decisiones. En la práctica, esto significa que cada evento relevante del ejercicio puede registrarse de forma consistente, cada cálculo puede conservarse con orden, y cada resultado puede analizarse posteriormente para retroalimentar la instrucción y fortalecer la operación del sistema.

A nivel de aprendizaje, el mayor aporte del curso fue consolidar una forma de trabajo rigurosa: partir del entendimiento del dominio (reglas de negocio y actores reales del entrenamiento), modelar con criterio, y recién después traducirlo a estructura física y programación en SQL Server. Esta metodología evitó improvisaciones y permitió sostener una arquitectura coherente, donde las entidades clave capturan tanto a los actores humanos como a los elementos técnicos del entrenamiento; además, la definición de atributos, claves y relaciones ayudó a eliminar redundancias, mantener integridad y preparar la base para crecer sin perder control.

En esa línea, el trabajo de modelado y validación demostró que la disciplina de diseño es lo que realmente "blinda" un sistema: las restricciones y reglas de integridad no se quedan como teoría, sino que se convierten en mecanismos que protegen la coherencia del simulador y obligan a que los datos reflejen la realidad operativa. Este proyecto me permitió entender que cada relación, clave foránea y constraint traduce una norma operativa y ayuda a que la base de datos funcione como un guardián de consistencia, precisión y trazabilidad.

Finalmente, me llevo la convicción de que este curso aporta directamente a la evolución de STACMO: una mejor gestión de datos no solo ordena la información, sino que habilita reportes más completos, estadísticas más precisas y una escalabilidad que proyecta el sistema hacia escenarios de mayor exigencia y adopción. En mi caso, esto tiene un valor especial porque el proyecto no nace como un ejercicio aislado, sino como un esfuerzo real que busca fortalecer capacidades y demostrar que la academia y la industria pueden integrarse para impulsar innovación tecnológica aplicable.

Gino Paolo Sassarini Bazán