Diseño lógico - Modelo relacional
El diseño lógico del sistema es la conversión directa del modelo conceptual (DER) a un modelo relacional que se puede implementar en SQL Server. En esta fase, las entidades se convierten en tablas relacionales, sus atributos se transforman en columnas con los tipos de datos adecuados, y las relaciones entre ellas se establecen mediante claves primarias y foráneas, siguiendo las reglas de cardinalidad y participación del modelo conceptual.

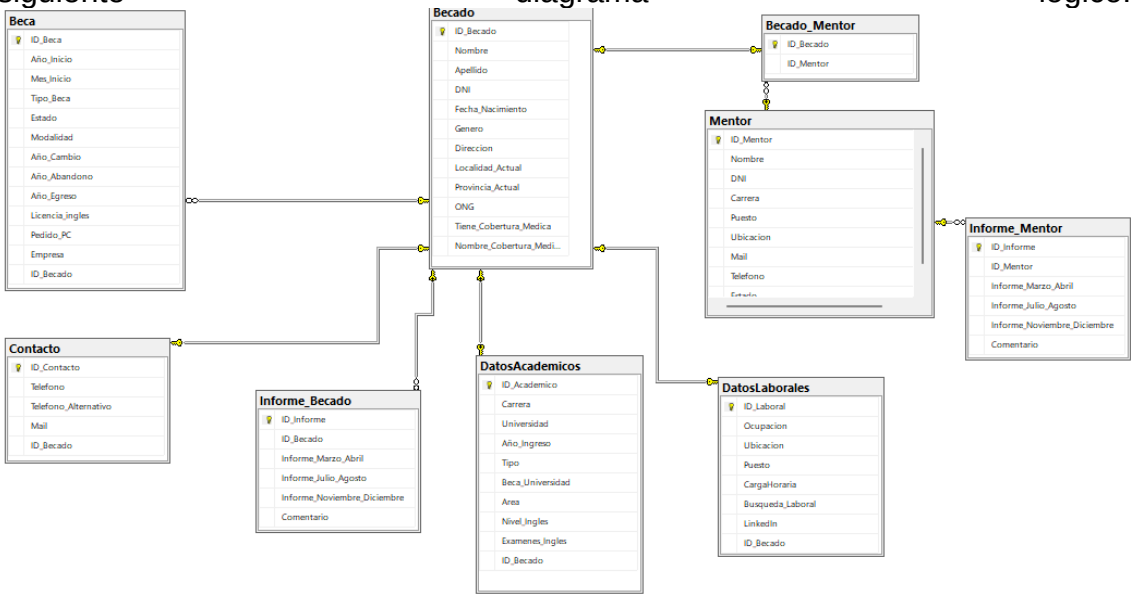
El modelo relacional resultante organiza de manera estructurada la información clave del sistema, distribuida en ocho áreas funcionales, garantizando la integridad referencial, la consistencia en el almacenamiento y una base sólida para futuras consultas y mantenimientos.

Este modelo presenta las siguientes características:

- **Claves primarias** que aseguran la unicidad de cada registro.
- **Claves foráneas** que mantienen la conexión lógica entre entidades relacionadas.
- Restricciones **CHECK** para validar los valores permitidos (como 'Sí' / 'No', o categorías específicas).
- Campos únicos (UNIQUE) en relaciones uno a uno.

Normalización hasta al menos la tercera forma normal (**3FN**), eliminando redundancias y asegurando que cada atributo dependa únicamente de la clave primaria.

El modelo relacional, diseñado en SQL Server, se ilustra gráficamente en el siguiente diagrama lógico:



Diseño físico - Implementación en SQL Server

El diseño físico representa la fase final del proceso de modelado de la base de datos, donde el modelo lógico es traducido directamente al sistema gestor de bases de datos elegido, en este caso SQL Server. En esta etapa, se crean las tablas, se asignan los tipos de datos más apropiados, se definen claves primarias y foráneas, restricciones de integridad y configuraciones necesarias para garantizar el correcto almacenamiento y funcionamiento del sistema.

El modelo relacional fue implementado mediante sentencias **CREATE TABLE** en SQL Server, respetando los siguientes criterios:

- Claves primarias (**PRIMARY KEY**) para identificar de forma única cada registro.
- Claves foráneas (**FOREIGN KEY**) para establecer las relaciones entre las entidades y asegurar la integridad referencial.
- Restricciones **CHECK** para validar valores permitidos en campos específicos, como los que aceptan solo 'Si' o 'No', o listas cerradas como 'Activo' / 'Inactivo'.
- Campos **UNIQUE** en relaciones uno a uno, como en Contacto, DatosAcademicos o DatosLaborales, que dependen exclusivamente de un becado.

Tipos de datos adecuados a la naturaleza del campo, por ejemplo:

- **VARCHAR**(n) para textos y cadenas alfanuméricas.
- **DATE** para fechas de nacimiento.
- INT con **IDENTITY** para generar identificadores automáticos en Becado.

Además, se crearon tablas intermedias, como **Becado_Mentor**, para modelar relaciones como "un becado puede o no tener un mentor" sin perder la flexibilidad del modelo ni comprometer la integridad de los datos.

Este diseño físico garantiza que la base de datos pueda ser implementada de forma directa y funcional, respetando las reglas definidas en el modelo conceptual y lógico, y facilitando tanto la carga de datos como las futuras consultas, informes y mantenimientos del sistema.