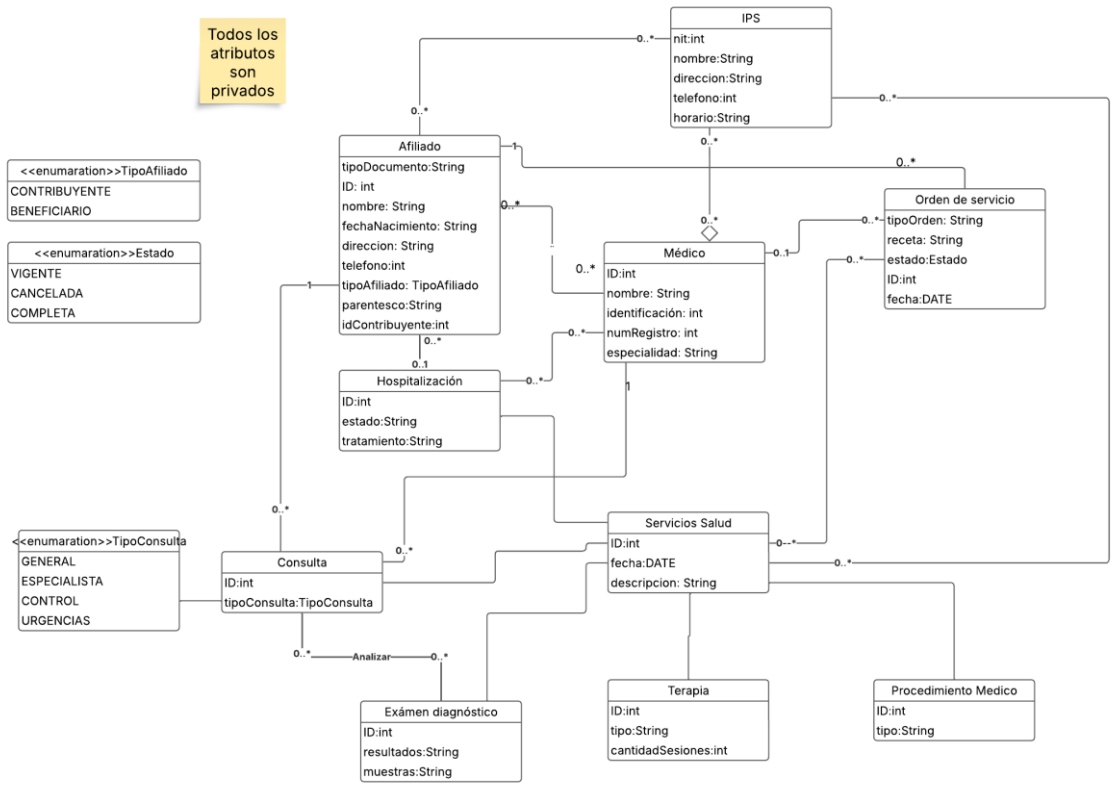
Entrega 1 – Parte 1
Proyecto Sistemas Transaccionales
ISIS 2304

1. Jeronimo Vasquez Ponce – 202223824
2. Paola Catherine Jimenez Jaque - 202116886
3. Yefran David Cespedes Cortes - 202316255

Modelo UML



1. Unificación y Reorganización de Clases
   a) Se unificaron las clases Consulta y ReservaConsulta, que anteriormente estaban definidas de forma separada, en una única entidad. La diferenciación se realiza mediante el atributo tipoConsulta, lo que permite gestionar de forma integrada tanto las reservas como las consultas, eliminando redundancias.
   b) De manera similar, se fusionaron las clases Consulta y ConsultaUrgencia en una sola clase, diferenciándose el tipo de consulta (urgencia o regular) a través de un atributo, lo que mejora la trazabilidad y simplifica el modelo.
   c) Las clases MédicoGeneral y Especialista se integraron en una única clase denominada Médico, que incorpora un atributo especialidad para determinar si el médico es general o especialista, eliminando duplicidades en las relaciones y facilitando la gestión de la información.

2. Ajustes en Relaciones y Cardinalidades
   a) La relación entre IPS y Médico se modificó: en el modelo anterior se planteaba una relación de agregación simple, mientras que en el nuevo UML se estableció una relación de asociación de muchos a muchos. Esto permite reflejar que un mismo médico puede trabajar en varias IPS, y cada IPS puede contar con múltiples médicos, lo que se alinea de forma precisa con la realidad del negocio.
   b) Se revisaron y ajustaron otras relaciones para evitar redundancias en los atributos; por ejemplo, se eliminó el atributo duplicado de tipo médico:String que coexistía con la relación establecida con la clase Médico, lo que simplifica el mantenimiento del modelo y evita inconsistencias.

3. Inclusión de Atributos y Enumeraciones Fundamentales
   a) En la entidad Orden de Servicio se añadió el atributo estado con valores restringidos (VIGENTE, CANCELADA, COMPLETA), lo que permite gestionar de manera precisa el ciclo de vida de las órdenes.
   b) En la entidad Afiliado se definió el atributo tipoAfiliado, restringido a los valores CONTRIBUYENTE o BENEFICIARIO, asegurando la correcta categorización de los usuarios.
   c) Se incorporaron enumeraciones y restricciones adicionales en otros atributos fundamentales, garantizando la integridad de los datos en el modelo relacional posterior.

4. mNuevas Clases de Servicios de Salud y Conexiones 1 a 1
   a) Se introdujo la nueva clase Servicios de Salud, que actúa como entidad central para representar los distintos servicios ofrecidos por la EPS.
   b) Además, se crearon clases adicionales que se conectan de forma 1 a 1 con Servicios de Salud para modelar de forma específica subprocesos o detalles asociados, como exámenes diagnósticos, terapias y procedimientos médicos. Estas conexiones 1 a 1 permiten una segmentación clara y modular de la información, facilitando la extensión del modelo y la implementación de restricciones específicas en el modelo relacional.

MODELO RELACIONAL

En el siguiente enlace se encuentra el modelo de datos relacional del proyecto, donde se detallan todas las tablas normalizadas hasta FNBC, junto sus claves primarias (PK) y claves foráneas (FK), garantizando una estructura óptima para el sistema:

PlantillaRelacional_Entrega1.5,BUENO.xlsx

El archivo de Excel contiene 18 tablas normalizadas hasta FNBC, con sus atributos organizados en columnas y especificando las claves primarias (PK) y las claves foráneas (FK) para garantizar la integridad referencial del sistema. A continuación, se presenta un resumen de las tablas incluidas en el modelo relacional de la base de datos:

AFILIADO

Esta tabla existe para almacenar la información personal de cada afiliado (por ejemplo, tipo de documento, nombre, fecha de nacimiento, dirección y, en su caso, información adicional que permita distinguir si es contribuyente o beneficiario). La clave primaria (por ejemplo, idAfiliado) determina de forma completa todos sus atributos, sin depender de partes de la clave ni de otros atributos; de ahí se cumple la BCNF.

MEDICO

La tabla MEDICO unifica la información de los médicos, incluyendo campos como identificación, nombre, número de registro y especialidad. Todos los atributos dependen íntegramente de la clave primaria (idMedico). Esto evita redundancias y garantiza que la entidad se encuentre normalizada hasta BCNF, reflejando el diseño del nuevo UML.

IPS

La tabla IPS representa a las Instituciones Prestadoras de Servicios de Salud, identificadas por un atributo único (por ejemplo, NIT). Incluye datos como nombre, dirección, teléfono y horario. Dado que cada atributo depende exclusivamente de la clave primaria, la tabla se encuentra en BCNF, cumpliendo con el modelo conceptual.

ORDEN_SERVICIO

Esta tabla registra las órdenes médicas emitidas, con atributos como tipo de orden, receta, estado (por ejemplo, VIGENTE, CANCELADA o COMPLETA) y fecha. Todos estos datos dependen de la clave primaria (idOrdenServicio), sin dependencias parciales ni transitivas, por lo que se encuentra normalizada en BCNF.

SERVICIO_SALUD

La tabla SERVICIO_SALUD actúa como entidad central para representar los servicios que ofrece la EPS. Atributos como descripción, fecha de disponibilidad y otros detalles dependen completamente de la clave primaria (idServicioSalud). Esto asegura que la tabla se encuentre en BCNF y centraliza la información de los servicios según el UML.

EXAMEN_DIAGNOSTICO

Esta tabla detalla los exámenes diagnósticos como un tipo específico de servicio de salud. Está vinculada de forma 1 a 1 con SERVICIO_SALUD, de modo que todos sus atributos (resultados, muestras, etc.) dependen completamente de su clave primaria (idExamenDiagnostico) o de la relación con la entidad principal, garantizando la normalización hasta BCNF.

TERAPIA

La tabla TERAPIA almacena información sobre los servicios de terapia, por ejemplo, el tipo de terapia y la cantidad de sesiones. Al depender cada atributo de la clave primaria (idTerapia) o de la relación con SERVICIO_SALUD (en un diseño 1 a 1), se asegura que no existan dependencias parciales ni transitivas, cumpliendo con BCNF.

PROCEDIMIENTO_MEDICO

Esta tabla se dedica a los procedimientos médicos especializados. Cada campo (como el tipo

de procedimiento) depende de la clave primaria (idProcedimientoMedico) o de la conexión con la entidad de servicio, garantizando la integridad de los datos y la normalización hasta BCNF.

CONSULTA

La tabla CONSULTA unifica la información relacionada con las consultas médicas (ya sean generales, de control o de urgencia). Los atributos (por ejemplo, diagnóstico, fecha, tipoConsulta) dependen íntegramente de la clave primaria (idConsulta). Esto elimina redundancias y asegura la conformidad con BCNF, reflejando la consolidación de consultas propuesta en el nuevo UML.

HOSPITALIZACION

La tabla HOSPITALIZACION registra los eventos de hospitalización, incluyendo atributos como fecha, tratamiento y estado. Todos estos dependen de la clave primaria (idHospitalizacion), asegurando que la tabla esté en BCNF, lo cual es esencial para una gestión precisa de estos eventos.

IPS_AFILIADO

Esta tabla intermedia gestiona la relación muchos a muchos entre IPS y AFILIADO, permitiendo asociar a los afiliados con las instituciones a las que pertenecen o con las que se relacionan. La clave compuesta (por ejemplo, nit de la IPS y idAfiliado) determina completamente el registro, garantizando la normalización hasta BCNF.

MEDICO_IPS

La tabla MEDICO_IPS es otra tabla intermedia, esta vez para reflejar la relación muchos a muchos entre MEDICO e IPS. Mediante una clave compuesta (idMedico y nit), se asegura que cada médico se pueda asociar a múltiples IPS y viceversa, y la dependencia completa de la clave compuesta confirma su normalización en BCNF.

CONSULTA_EXAMEN

La tabla CONSULTA_EXAMEN relaciona las consultas con los exámenes diagnósticos realizados. La clave compuesta (por ejemplo, idConsulta e idExamen) garantiza que la asignación sea única y que cada atributo dependa de la totalidad de la clave, lo que se alinea con los criterios de BCNF.

HOSPITALIZACION_MEDICO

Esta tabla relaciona hospitalizaciones con los médicos involucrados (por ejemplo, el médico que atiende la hospitalización). Se utiliza una clave compuesta (idHospitalizacion y idMedico) para garantizar la unicidad, y todos los atributos dependen completamente de dicha clave, asegurando la normalización hasta BCNF.

AFILIADO_HOSPITALIZACION

La tabla AFILIADO_HOSPITALIZACION gestiona la relación entre los afiliados y los eventos de hospitalización, permitiendo registrar qué afiliado participó en cada hospitalización. Con una clave compuesta (idAfiliado e idHospitalizacion), se garantiza que cada relación es única y normalizada según BCNF.

IPS_SERVICIO

La tabla IPS_SERVICIO normaliza la relación entre una IPS y los servicios de salud que ofrece. Emplea una clave compuesta (por ejemplo, nit e idServicioSalud) para asegurar que cada asignación es única, y todos los atributos dependen completamente de esa clave, lo que confirma su conformidad con BCNF.

ORDEN_SERVICIO_SERVICIOS

Esta tabla permite asociar una orden de servicio con uno o varios servicios de salud. Utiliza una clave compuesta (idOrdenServicio e idServicioSalud) que garantiza que cada asociación es única, y al depender de la totalidad de la clave compuesta, se encuentra normalizada en BCNF.

MEDICO_AFILIADO

La tabla MEDICO_AFILIADO relaciona a los médicos con los afiliados, reflejando situaciones en las que un médico atiende a uno o varios afiliados y, de ser necesario, viceversa. La clave compuesta (idMedico e idAfiliado) asegura la unicidad de cada vínculo, y la dependencia funcional completa de esta clave garantiza que la tabla se encuentra en BCNF.

Creación base de datos, scripts de la creación de cada tabla de la BD.

```
CREATE TABLE Afiliados (
    idAfiliado NUMBER GENERATED BY DEFAULT AS IDENTITY,
    tipoDocumento VARCHAR(50) NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    fechaNacimiento DATE NOT NULL,
    direccion VARCHAR(250) NOT NULL,
    telefono VARCHAR(15) NOT NULL,
    parentesco VARCHAR(50) NOT NULL,
    tipoAfiliado VARCHAR(50) NOT NULL CHECK (tipoAfiliado IN ('CONTRIBUYENTE','BENEFICIARIO')),
    idContribuyente NUMBER NULL,
    CONSTRAINT pk_Afiliado PRIMARY KEY (idAfiliado),
    FOREIGN KEY (idContribuyente) REFERENCES Afiliados(idAfiliado) ON DELETE SET NULL
);
```

Esta tabla almacena la información personal de cada afiliado, como tipo de documento, nombre, fecha de nacimiento, dirección, teléfono, parentesco y tipo de afiliado. Además, permite referenciar a otro afiliado como contribuyente en caso de beneficiario, manteniendo la integridad referencial.

```sql
CREATE TABLE IPSs(
    nit NUMBER GENERATED BY DEFAULT AS IDENTITY,
    nombre VARCHAR(100) NOT NULL,
    direccion VARCHAR(255) NOT NULL,
    telefono VARCHAR(20) NOT NULL,
    horario VARCHAR(50) NOT NULL,
    CONSTRAINT pk_ips PRIMARY KEY (nit)
);
```

Esta tabla representa las Instituciones Prestadoras de Servicios de Salud, identificadas por un NIT único. Se guardan atributos como nombre, dirección, teléfono y horario, lo que permite identificar y gestionar cada IPS de forma independiente.

```sql
CREATE TABLE Medicos(
    idMedico NUMBER GENERATED BY DEFAULT AS IDENTITY,
    identificacion VARCHAR(50) NOT NULL UNIQUE,
    nombre VARCHAR(100) NOT NULL,
    numRegistro VARCHAR(50) NOT NULL UNIQUE,
    especialidad VARCHAR(100) NOT NULL,
    CONSTRAINT pk_medico PRIMARY KEY(idMedico)
);
```

Esta tabla unifica la información de los médicos, almacenando datos como identificación, nombre, número de registro y especialidad. Las restricciones de unicidad en ciertos campos aseguran la integridad y evitan duplicaciones.

```sql
CREATE TABLE OrdenServicios(
    idOrden NUMBER GENERATED BY DEFAULT AS IDENTITY,
    tipoOrden VARCHAR(250) NOT NULL,
    receta VARCHAR(1000) NOT NULL,
    estado VARCHAR(150) NOT NULL CHECK(estado IN ('VIGENTE','CANCELADA','COMPLETADA')),
    fecha DATE NOT NULL,
    idMedico NUMBER NOT NULL,
    idAfiliado NUMBER NOT NULL,
    CONSTRAINT pk_ordenServicio PRIMARY KEY(idOrden),
    FOREIGN KEY (idMedico) REFERENCES Medicos(idMedico),
    FOREIGN KEY (idAfiliado) REFERENCES Afiliados(idAfiliado)
);
```

Esta tabla registra las órdenes médicas, incluyendo detalles como tipo de orden, receta, estado (con valores restringidos a 'VIGENTE', 'CANCELADA' o 'COMPLETADA') y fecha. Además, asocia cada orden con un médico y un afiliado mediante claves foráneas.

```sql
CREATE TABLE ServiciosSalud (
    idServicio NUMBER GENERATED BY DEFAULT AS IDENTITY,
    fecha DATE NOT NULL,
    descripcion VARCHAR2(255) NOT NULL,
    CONSTRAINT pk_servicio PRIMARY KEY(idServicio)
);
```

Esta tabla actúa como la entidad central para los servicios que ofrece la EPS. Contiene la fecha y una descripción de cada servicio, permitiendo centralizar y gestionar de forma uniforme los distintos tipos de servicios.

```sql
CREATE TABLE ExamenesDiagnosticos(
    idExamen NUMBER GENERATED BY DEFAULT AS IDENTITY,
    resultados VARCHAR(2000) NOT NULL,
    muestras VARCHAR(2000) NOT NULL,
    idServicio NUMBER NOT NULL,
    CONSTRAINT pk_examen PRIMARY KEY (idExamen),
    FOREIGN KEY(idServicio) REFERENCES ServiciosSalud(idServicio) ON DELETE CASCADE
);
```

Esta tabla detalla los exámenes diagnósticos, almacenando resultados y muestras. Se relaciona con un servicio de salud, de forma que la eliminación de un servicio también elimina sus exámenes asociados.

```sql
CREATE TABLE Consultas(
    idConsulta NUMBER GENERATED BY DEFAULT AS IDENTITY,
    tipoConsulta VARCHAR(100) NOT NULL CHECK (tipoConsulta IN ('GENERAL','ESPECIALISTA','URGENCIA','CONTROL')),
    idAfiliado NUMBER NOT NULL,
    idMedico NUMBER NULL,
    idServicio NUMBER NOT NULL,
    CONSTRAINT pk_consulta_ PRIMARY KEY (idConsulta),
    FOREIGN KEY (idMedico) REFERENCES Medicos(idMedico) ON DELETE CASCADE,
    FOREIGN KEY (idAfiliado) REFERENCES Afiliados(idAfiliado) ON DELETE CASCADE,
    FOREIGN KEY(idServicio) REFERENCES ServiciosSalud(idServicio) ON DELETE CASCADE
);
```

Esta tabla unifica la información de las consultas médicas, distinguiendo el tipo de consulta mediante un campo que admite valores como 'GENERAL', 'ESPECIALISTA', 'URGENCIA' o 'CONTROL'. Relaciona cada consulta con un afiliado, y opcionalmente con un médico y un servicio de salud.

```
CREATE TABLE Terapias(
    idTerapia NUMBER GENERATED BY DEFAULT AS IDENTITY,
    tipo VARCHAR(250) NOT NULL,
    cantidadSesiones NUMBER NOT NULL,
    idServicio NUMBER NOT NULL,
    CONSTRAINT pk_terapia PRIMARY KEY (idTerapia),
    FOREIGN KEY(idServicio) REFERENCES ServiciosSalud(idServicio) ON DELETE CASCADE
);
```

Esta tabla almacena información sobre los servicios de terapia, indicando el tipo de terapia y la cantidad de sesiones programadas. Cada terapia se asocia a un servicio de salud para su correcta gestión.

```
  CREATE TABLE ProcedimientosMedicos(
      idProcedimiento NUMBER GENERATED BY DEFAULT AS IDENTITY,
      tipo VARCHAR(250) NOT NULL,
      idServicio NUMBER NOT NULL,
      CONSTRAINT pk_procedimiento PRIMARY KEY (idProcedimiento),
      FOREIGN KEY(idServicio) REFERENCES ServiciosSalud(idServicio) ON DELETE CASCADE
  );
```

Esta tabla contiene los datos de los procedimientos médicos especializados, como el tipo de procedimiento. Se vincula a un servicio de salud, permitiendo una gestión específica y detallada de estos procedimientos

```
CREATE TABLE Hospitalizaciones(
    idHospitalizacion NUMBER GENERATED BY DEFAULT AS IDENTITY,
    estado VARCHAR(255) NOT NULL,
    tratamiento VARCHAR(2000) NOT NULL,
    idServicio NUMBER NOT NULL,
    CONSTRAINT pk_hospitalizacion PRIMARY KEY (idHospitalizacion),
    FOREIGN KEY(idServicio) REFERENCES ServiciosSalud(idServicio) ON DELETE CASCADE
);
```

Esta tabla registra los eventos de hospitalización, incluyendo información sobre el estado y tratamiento. Se asocia a un servicio de salud, lo que permite gestionar y rastrear cada hospitalización de manera individual.

```
CREATE TABLE Afiliado_IPS(
    idAfiliado NUMBER NOT NULL,
    nit NUMBER NOT NULL,
    PRIMARY KEY (idAfiliado,nit),
    FOREIGN KEY (idAfiliado) REFERENCES Afiliados(idAfiliado) ON DELETE CASCADE,
    FOREIGN KEY (nit) REFERENCES IPSs(nit) ON DELETE CASCADE
);
```

Es una tabla intermedia que gestiona la relación muchos a muchos entre afiliados e IPSs.

Permite asociar a cada afiliado con una o más IPS, garantizando la integridad referencial mediante una clave compuesta.

```
CREATE TABLE Medico_IPS(
    idMedico NUMBER NOT NULL,
    nit NUMBER NOT NULL,
    PRIMARY KEY (idMedico,nit),
    FOREIGN KEY (idMedico) REFERENCES Medicos(idMedico) ON DELETE CASCADE,
    FOREIGN KEY (nit) REFERENCES IPSs(nit) ON DELETE CASCADE
);
```

Esta tabla gestiona la relación muchos a muchos entre médicos e IPSs, permitiendo que un mismo médico se asocie a varias IPS. La clave compuesta asegura que cada vínculo es único y consistente.

```
CREATE TABLE Consulta_Examen(

    idConsulta NUMBER NOT NULL,
    idExamen NUMBER NOT NULL,
    PRIMARY KEY (idConsulta,idExamen),
    FOREIGN KEY (idConsulta) REFERENCES Consultas(idConsulta) ON DELETE CASCADE,
    FOREIGN KEY (idExamen) REFERENCES ExamenesDiagnosticos(idExamen) ON DELETE CASCADE
);
```

Esta tabla vincula las consultas con los exámenes diagnósticos realizados, permitiendo que una consulta se asocie a uno o varios exámenes. La tabla utiliza una clave compuesta para garantizar la unicidad de cada asociación.

```
CREATE TABLE Hospitalizacion_Medico(
    idHospitalizacion NUMBER NOT NULL,
    idMedico NUMBER NOT NULL,
    PRIMARY KEY (idHospitalizacion, idMedico),
    FOREIGN KEY (idMedico) REFERENCES Medicos(idMedico) ON DELETE CASCADE,
    FOREIGN KEY (idHospitalizacion) REFERENCES Hospitalizaciones(idHospitalizacion) ON DELETE CASCADE
);
```

Esta tabla asocia cada hospitalización con los médicos que participan en su atención. Utiliza una clave compuesta para asegurar que cada relación es única y se mantiene la integridad referencial.

```
CREATE TABLE Hospitalizacion_Afiliado(
    idHospitalizacion NUMBER NOT NULL,
    idAfiliado NUMBER NOT NULL,
    PRIMARY KEY (idHospitalizacion, idAfiliado),
    FOREIGN KEY (idAfiliado) REFERENCES Afiliados(idAfiliado) ON DELETE CASCADE,
    FOREIGN KEY (idHospitalizacion) REFERENCES Hospitalizaciones(idHospitalizacion) ON DELETE CASCADE
);
```

Esta tabla registra la relación entre hospitalizaciones y afiliados, permitiendo identificar qué afiliados están involucrados en cada evento de hospitalización. La clave compuesta garantiza la unicidad de cada vínculo.

```
CREATE TABLE Servicio_IPS(
    idServicio NUMBER NOT NULL,
    nit NUMBER NOT NULL,
    PRIMARY KEY (idServicio,nit),
    FOREIGN KEY (idServicio) REFERENCES ServiciosSalud(idServicio) ON DELETE CASCADE,
    FOREIGN KEY (nit) REFERENCES IPSs(nit) ON DELETE CASCADE
);
```

Esta tabla vincula los servicios de salud ofrecidos con las IPS que los proporcionan. La clave compuesta permite asociar cada servicio a una o más IPS, facilitando la gestión de estas relaciones.

```
CREATE TABLE Orden_Servicio(

    idOrden NUMBER NOT NULL,
    idServicio NUMBER NOT NULL,
    PRIMARY KEY(idOrden,idServicio),
    FOREIGN KEY (idOrden) REFERENCES OrdenServicios(idOrden) ON DELETE CASCADE,
    FOREIGN KEY (idServicio) REFERENCES ServiciosSalud(idServicio) ON DELETE CASCADE
);
```

Esta tabla asocia cada orden médica con los servicios de salud prescritos. La tabla utiliza una clave compuesta para asegurar que la relación entre una orden y sus servicios es única y consistente.

```
CREATE TABLE Medico_Afiliado(
    idMedico NUMBER NOT NULL,
    idAfiliado NUMBER NOT NULL,
    PRIMARY KEY (idMedico,idAfiliado),
    FOREIGN KEY (idMedico) REFERENCES Medicos(idMedico) ON DELETE CASCADE,
    FOREIGN KEY (idAfiliado) REFERENCES Afiliados(idAfiliado) ON DELETE CASCADE

);
```

Esta tabla gestiona la relación muchos a muchos entre médicos y afiliados, permitiendo que se registre qué médicos atienden a qué afiliados. La clave compuesta garantiza la integridad y

unicidad de cada asignación.

Documentación de las clases de Java Spring y de la arquitectura de la aplicación

La aplicación está construida sobre el framework Spring Boot y se basa en una arquitectura MVC en capas que facilita la separación de responsabilidades. En la primera capa se encuentran las entidades, que son clases Java mapeadas a tablas en la base de datos. Cada una de estas clases contiene atributos que reflejan las columnas correspondientes, además de los métodos necesarios para la manipulación de esos datos. El uso de anotaciones de JPA, como @Entity y @Table, permite que el framework gestione la persistencia de forma automática, asignando un identificador único a cada registro y vinculando cada atributo con su respectiva columna.

En la segunda capa se ubican los repositorios, los cuales sirven como intermediarios entre las entidades y la base de datos. Estas interfaces o clases utilizan anotaciones como @Repository y, en muchos casos, extienden de JpaRepository o incluyen consultas nativas mediante la anotación @Query. Su función principal es proveer métodos de creación, lectura, actualización y eliminación de datos, manejando las transacciones de forma transparente y asegurando la integridad referencial. Al abstraer el acceso a la base de datos, los repositorios evitan que otras partes de la aplicación deban conocer la lógica interna de las operaciones SQL.

Finalmente, la tercera capa la componen los controladores, anotados como @RestController o @Controller. Estos exponen endpoints HTTP que permiten a los clientes (por ejemplo, Postman) interactuar con el sistema. Cada controlador inyecta los repositorios que necesita y define métodos que atienden solicitudes específicas, procesan los datos de entrada y devuelven respuestas en formato JSON o en vistas, dependiendo de la configuración del proyecto. De esta manera, la lógica de negocio se concentra en estos controladores y en los servicios asociados, mientras que los repositorios se encargan de la persistencia y las entidades representan la estructura de la información.

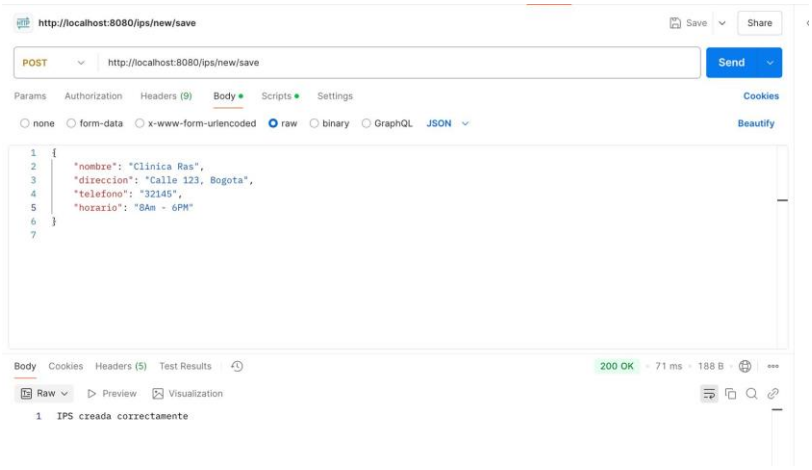
Documentación de las consultas RFC
RF1:

El RF1 se encarga de registrar una nueva IPS en el sistema, lo que implica insertar un registro en la tabla correspondiente con los datos esenciales de la institución de salud, como nombre, dirección, teléfono y horario. Este proceso se inicia al enviar una solicitud desde Postman al endpoint designado, que recibe la información en formato JSON y la remite al repositorio encargado de persistirla en la base de datos.

El repositorio para IPS, definido en la clase IpsRepository, extiende JpaRepository y utiliza consultas nativas para realizar operaciones específicas. En este código, el método insertarIp ejecuta un INSERT en la tabla IPSS, aprovechando una secuencia para generar automáticamente el identificador único (nit) de la IPS. Además, el repositorio define métodos como darIps para obtener todas las IPS, darIp para consultar una IPS en particular por su nit, y métodos para actualizar y eliminar registros. Las anotaciones @Modifying y

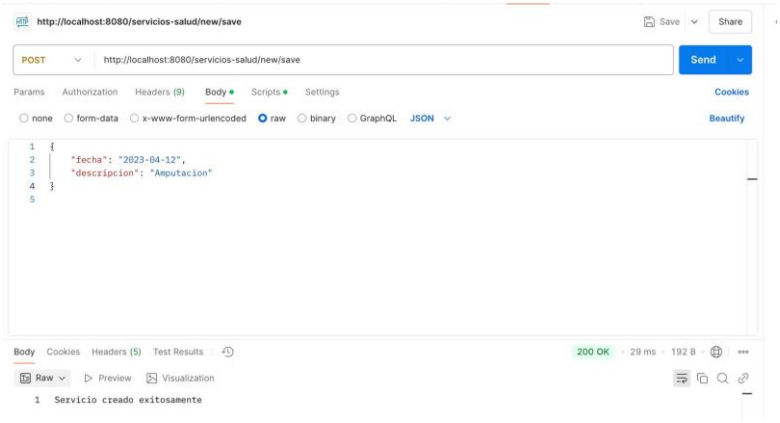
@Transactional garantizan que las operaciones de inserción, actualización y eliminación se ejecuten de forma atómica y segura, permitiendo manejar los cambios en la base de datos de manera consistente.

Para probar este RF en Postman, se envía una solicitud POST al endpoint asociado con la creación de IPS, incluyendo en el cuerpo de la petición los campos requeridos (nombre, dirección, teléfono y horario). El controlador recibe esta información y llama al método insertarIp del repositorio, lo que resulta en la inserción del nuevo registro en la base de datos. El mensaje de respuesta confirma que la IPS se ha registrado correctamente, y los datos pueden verse reflejados en la base de datos, como se ilustra en la imagen:



RF2: El RF2 se encarga de registrar un servicio de salud que la EPS ofrece a sus afiliados, lo que implica insertar un nuevo registro en la tabla correspondiente con datos esenciales como la fecha en la que se brinda el servicio y una descripción detallada del mismo. Este proceso se activa al enviar una solicitud POST desde Postman al endpoint designado para registrar servicios, donde se recibe la información en formato JSON. El controlador valida que los campos obligatorios estén presentes y realiza las conversiones necesarias, en particular transformando la fecha al formato requerido, para luego remitir los datos al repositorio que se encargará de persistir la información en la base de datos.

El repositorio de ServicioSalud, definido en la clase ServicioSaludRepository, extiende de JpaRepository y utiliza consultas nativas para realizar las operaciones de inserción, actualización, eliminación y consulta sobre la tabla ServiciosSalud. En este caso, el método insertarServicioSalud ejecuta una sentencia INSERT que utiliza la secuencia SERVICIO_SEQ.nextval para generar automáticamente el identificador único (idServicio) del servicio, y emplea la función TO_DATE para convertir la fecha al formato 'YYYY-MM-DD'. Las anotaciones @Modifying y @Transactional garantizan que la operación se ejecute de forma atómica y segura. Al probar este RF en Postman se envía una solicitud con la fecha y la descripción, el controlador invoca el método correspondiente del repositorio y se inserta el registro en la base de datos, tal como se observa en la imagen:

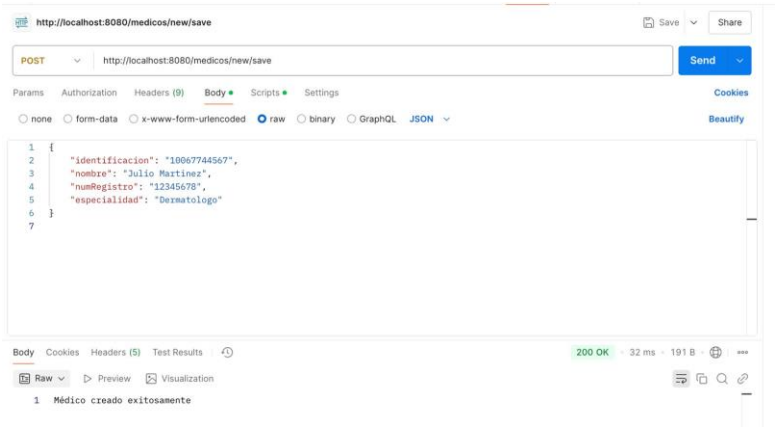


RF3: En el RF3 se busca asignar un servicio de salud a una IPS, lo que implica crear y registrar la relación entre un servicio ya existente y la institución que lo ofrece. Esta operación permite determinar qué servicios se ofrecen en cada IPS, lo que es fundamental para la gestión y consulta posterior de los servicios disponibles. La acción se realiza registrando una nueva asociación en la tabla intermedia correspondiente, lo que garantiza que cada asignación se realice de manera controlada y única.

El repositorio que actúa en el RF3 es el IpsServicioRepository, el cual extiende JpaRepository y utiliza consultas nativas para insertar, actualizar, eliminar y consultar las asociaciones entre servicios y IPSs. Mediante el uso de anotaciones como @Modifying y @Transactional, este repositorio asegura que la operación de asignación se ejecute de forma atómica y consistente, respetando las claves foráneas y las restricciones definidas en la base de datos. La integración de este componente permite que, al recibir los datos del servicio y la IPS a través del controlador, la relación se establezca de manera fiable en el sistema.

RF4: El RF4 se encarga de registrar un nuevo médico en el sistema, permitiendo que la EPS asocie a un profesional de la salud con los servicios que presta. Este proceso se activa mediante una solicitud POST al endpoint correspondiente, donde se envían los datos del médico en formato JSON. La información incluye la identificación, nombre, número de registro y especialidad, la cual es validada en el controlador para asegurarse de que todos los campos obligatorios están presentes antes de proceder a su inserción.

El repositorio de médicos, definido en la clase MedicoRepository, gestiona el acceso a la tabla Medicos a través de consultas nativas. En particular, el método insertarMedico realiza un INSERT utilizando la secuencia MEDICO_SEQ.nextval para generar automáticamente el identificador único del médico, y asegura que los datos se inserten de forma consistente en la base de datos. El controlador de médicos (MedicosController) invoca este método al recibir la solicitud de registro, y se encarga de enviar una respuesta confirmando que el médico se ha creado exitosamente. De esta forma, al probar el RF4 con Postman se observa que el registro se realiza correctamente en la base de datos, como se ilustra en la imagen:



RF5: El RF5 se encarga de registrar un nuevo afiliado en el sistema, capturando toda la información personal necesaria, como el tipo de documento, nombre, fecha de nacimiento, dirección, teléfono, parentesco y el tipo de afiliación (CONTRIBUYENTE o BENEFICIARIO). Este proceso es esencial para garantizar que cada usuario quede debidamente identificado y para establecer las relaciones correspondientes con otros registros, como los beneficiarios que dependen de un contribuyente. Así, la inserción de un nuevo afiliado alimenta la base de datos con registros fundamentales para el funcionamiento de la aplicación.

Para este RF, el AfiliadoRepository es el encargado de gestionar la persistencia de los datos del afiliado. Este repositorio utiliza consultas nativas a través de la anotación @Query para insertar, actualizar, eliminar y recuperar registros de la tabla Afiliados. Con el soporte de anotaciones @Modifying y @Transactional, el AfiliadoRepository garantiza que las operaciones se realicen de forma segura y en el contexto de una transacción, cumpliendo con las restricciones de integridad y las reglas de negocio establecidas en el modelo.

RF6: El RF6 se centra en registrar una orden de servicio de salud emitida por un médico para un afiliado. Este proceso implica la creación de un nuevo registro en la tabla de órdenes (OrdenServicios), donde se capturan datos críticos como el tipo de orden, la receta, el estado de la orden, la fecha y las referencias tanto al médico como al afiliado. La operación formaliza la prescripción de un servicio y establece las bases para posteriores procesos, como la asignación y ejecución del servicio.

El OrdenServicioRepository es el componente responsable en este RF, al proporcionar métodos para insertar, actualizar, eliminar y consultar órdenes de servicio mediante consultas nativas. Además, el OrdenServicioServiciosRepository se utiliza para vincular cada orden con uno o más servicios específicos. Juntos, estos repositorios se aseguran de que cada orden se registre cumpliendo las restricciones de integridad referencial y que las relaciones con los registros de médicos, afiliados y servicios se establezcan de forma coherente y transaccional.

RF7: El RF7 tiene por objetivo agendar un servicio de salud para un afiliado, lo que implica reservar una cita o generar una consulta en el sistema. Este proceso consiste en registrar en la

base de datos la intención de recibir el servicio en una fecha determinada, permitiendo coordinar la disponibilidad de los profesionales y las IPS, así como gestionar la planificación de los recursos necesarios. De esta forma, se habilita la operación de reserva o agendamiento, que es vital para la organización y el flujo de atención de la EPS.

El componente encargado de esta operación es el ConsultaRepository, que administra los registros de consultas en la tabla Consultas. Este repositorio define métodos mediante consultas nativas para insertar, actualizar, eliminar y consultar las reservas de servicio (consultas) realizadas por los afiliados. Al trabajar en conjunto con otros repositorios que gestionan datos de afiliados y médicos, el ConsultaRepository garantiza que el agendamiento se realice en un contexto transaccional, respetando las relaciones y restricciones definidas en el modelo.

RF8: Finalmente, el RF8 se encarga de registrar la prestación efectiva de un servicio de salud a un afiliado por parte de una IPS. Esta operación documenta la ejecución del servicio, incluyendo aspectos como la fecha, el estado y el tratamiento o procedimiento realizado, y establece la relación entre la prestación del servicio y las entidades involucradas (afiliados, IPS, médicos, etc.). Este registro es esencial para el seguimiento y la evaluación de la atención prestada, permitiendo auditar y controlar el uso de los recursos del sistema.

Para cumplir con el RF8 se utilizan varios repositorios coordinados: el AfiliadoHospitalizacionRepository registra la relación entre afiliados y hospitalizaciones, mientras que el HospitalizacionMedicoRepository se encarga de asociar la hospitalización con el médico que la atendió. Además, el IpsAfiliadoRepository y el OrdenServicioRepository pueden participar en la vinculación de los datos de prestación del servicio. Estos repositorios, mediante consultas nativas y métodos transaccionales, aseguran que la prestación del servicio se registre de forma íntegra y consistente, cumpliendo con las restricciones de integridad referencial y las reglas de negocio definidas en el sistema.

Documentación archivos .sql para los RF 1, 2, 3 y 4.

RF 1 Registrar IPS

```
CREATE OR REPLACE PROCEDURE registrarIPS(
    p_nombre     IN VARCHAR2,
    p_direccion  IN VARCHAR2,
    p_telefono   IN VARCHAR2,
    p_horario    IN VARCHAR2,
    p_nit        OUT NUMBER
) AS
BEGIN
    INSERT INTO IPSs (nombre, direccion, telefono, horario)
    VALUES (p_nombre, p_direccion, p_telefono, p_horario)
    RETURNING nit INTO p_nit;
    COMMIT;

    END;
/
```

El código de RF1 define el procedimiento almacenado registrarIPS, el cual se encarga de insertar un nuevo registro en la tabla IPSs utilizando los parámetros de nombre, dirección, teléfono y horario que se le suministran. Este procedimiento utiliza la cláusula RETURNING para capturar y devolver el NIT generado automáticamente, asegurando que el registro se haya creado de forma exitosa. Además, se maneja la transacción con COMMIT

RF2 Registrar servicio de salud

```
CREATE OR REPLACE PROCEDURE registrarServicioSalud(
    p_fecha       IN DATE,
    p_descripcion IN VARCHAR2,
    p_idServicio  OUT NUMBER
) AS
BEGIN
    INSERT INTO ServiciosSalud (fecha, descripcion)
    VALUES (p_fecha, p_descripcion)
    RETURNING idServicio INTO p_idServicio;
    COMMIT;

    END;
/
```

El procedimiento registrarServicioSalud inserta un registro en la tabla ServiciosSalud, tomando como parámetros la fecha y la descripción del servicio. Al igual que en RF1, utiliza la cláusula RETURNING para obtener el identificador único del servicio insertado

(idServicio), permitiendo su posterior uso en otras relaciones. Se aplica un COMMIT al finalizar la inserción para confirmar la transacción

RF3 Asignar un servicio de salud a una IPS

```
CREATE OR REPLACE PROCEDURE asignarServicioIPS(
    p_idServicio IN NUMBER,
    p_nitIPS     IN NUMBER
) AS
BEGIN
    INSERT INTO Servicio_IPS (idServicio, nit)
    VALUES (p_idServicio, p_nitIPS);
    COMMIT;

    END;
/
```

El procedimiento asignarServicioIPS se encarga de crear la asociación entre un servicio de salud y una IPS, insertando un registro en la tabla intermedia Servicio_IPS. Recibe como parámetros el id del servicio y el NIT de la IPS, y utiliza estos valores para establecer la relación entre ambas entidades. La combinación de estos valores es única y, al ejecutarse la sentencia de inserción, se confirma la transacción con COMMIT

RF4 Registrar Medico

```
CREATE OR REPLACE PROCEDURE registrarMedico(
    p_identificacion IN VARCHAR2,
    p_nombre         IN VARCHAR2,
    p_numRegistro    IN VARCHAR2,
    p_especialidad   IN VARCHAR2,
    p_nitIPS         IN NUMBER,
    p_idMedico       OUT NUMBER
) AS
BEGIN
    INSERT INTO Medicos (identificacion, nombre, numRegistro, especialidad)
    VALUES (p_identificacion, p_nombre, p_numRegistro, p_especialidad)
    RETURNING idMedico INTO p_idMedico;

    INSERT INTO Medico_IPS (idMedico, nit)
    VALUES (p_idMedico, p_nitIPS);

    COMMIT;

    END;
/
```

El procedimiento registrarMedico inserta un nuevo registro en la tabla Medicos con los datos del médico (identificación, nombre, número de registro y especialidad). Una vez creado el médico, el procedimiento utiliza el identificador generado para asociarlo a una IPS en la tabla intermedia Medico_IPS, estableciendo la relación entre el médico y la institución de salud correspondiente. Se utilizan las cláusulas RETURNING y COMMIT para garantizar que ambas inserciones se realicen de forma atómica.

Cuando se ejecutan las consultas SQL para los RF, se crean los registros correspondientes en las tablas de la base de datos. Estos datos se pueden visualizar utilizando una consulta SELECT o mediante la salida en SQL Developer, mostrando la información insertada tal como se ilustra en la imagen

```
Procedure REGISTRARIPS compilado

RF1: IPS registrada con NIT: 1

Procedimiento PL/SQL terminado correctamente.

Procedure REGISTRARSERVICIOSALUD compilado

RF2: Servicio registrado con ID: 1

Procedimiento PL/SQL terminado correctamente.

Procedure ASIGNARSERVICIOIPS compilado

RF3: Servicio 1 asignado a la IPS con NIT: 1

Procedimiento PL/SQL terminado correctamente.

Procedure REGISTRARMEDICO compilado

RF4: Médico registrado con ID: 1

Procedimiento PL/SQL terminado correctamente.

Procedure REGISTRARIPS compilado

RF1: IPS registrada con NIT: 2
```

Escenarios de pruebas:

```sql
-----------------------------------------------------
-- Escenario 1: Prueba de unicidad de tuplas en IPSs
-----------------------------------------------------

INSERT INTO IPSs (nit, nombre, direccion, telefono, horario)
VALUES (100, 'IPS 1', 'D', '123456789', 'L-V 8am-6pm');
DBMS_OUTPUT.PUT_LINE('Escenario 1: Primera insercion de IPS con nit=100 exitosa.');


BEGIN
    INSERT INTO IPSs (nit, nombre, direccion, telefono, horario)
    VALUES (100, 'IPS 1-1', 'DD', '987654321', 'L-V 9am-5pm');
    DBMS_OUTPUT.PUT_LINE('Escenario 1: Segunda insercion con nit=100 exitosa (esto no deberia ocurrir).');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Escenario 1: Error de unicidad: ' || SQLERRM);
END;
/
```

```
-----------------------------------------------------
-- Escenario 2: Prueba de integridad referencial con FK en Afiliado_IPS
-----------------------------------------------------

BEGIN
    INSERT INTO Afiliado_IPS (idAfiliado, nit) VALUES (1, 1);
    DBMS_OUTPUT.PUT_LINE('Escenario 2: Insercion con FK válida en Afiliado_IPS (idAfiliado=1, nit=1) exitosa.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Escenario 2: Error en inserción FK valida en Afiliado_IPS: ' || SQLERRM);
END;
/


BEGIN
    INSERT INTO Afiliado_IPS (idAfiliado, nit) VALUES (999999999, 1);
    DBMS_OUTPUT.PUT_LINE('Escenario 2: Insercion con FK invalida (idAfiliado=999, nit=1) exitosa (esto no deberia ocurrir).')
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Escenario 2: Error de integridad referencial capturado (idAfiliado=999, nit=1): ' || SQLERRM);
END;
/
```

```
-----------------------------------------------------
-- Escenario 3: Prueba de integridad de restricciones de chequeo en Afiliados
-----------------------------------------------------

BEGIN
    INSERT INTO Afiliados (idAfiliado, tipoDocumento, nombre, fechaNacimiento, direccion, telefono, parentesco, tipoAfiliado)
    VALUES (999, 'CC', 'Pepito Perez', DATE '2000-01-01', 'D', '3000000000', 'Prueba', 'INVALIDO');
    DBMS_OUTPUT.PUT_LINE('Escenario 3: Insercion en Afiliados con tipoAfiliado invalido exitosa (esto no debería ocurrir).');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Escenario 3: Error de restriccion de chequeo capturado: ' || SQLERRM);
END;
/
```

```
Error que empieza en la linea: 186 del comando :
INSERT INTO IPSs (nit, nombre, direccion, telefono, horario)
VALUES (100, 'IPS 1', 'D', '123456789', 'L-V 8am-6pm')
Informe de error -
ORA-00001: restricción única (ISIS2304D25202510.PK_IPS) violada

https://docs.oracle.com/error-help/db/ora-00001/

More Details :
https://docs.oracle.com/error-help/db/ora-00001/

Error que empieza en la linea: 188 del comando :
DBMS_OUTPUT.PUT_LINE('Escenario 1: Primera inserción de IPS con nit=100 exitosa.')
Informe de error -
Comando desconocido

Escenario 1: Error de unicidad: ORA-00001: restricción única (ISIS2304D25202510.PK_IPS) violada

Procedimiento PL/SQL terminado correctamente.
Escenario 2: Error en inserción FK válida en Afiliado_IPS: ORA-00001: restricción única (ISIS2304D25202510.SYS_C001439344) violada

Procedimiento PL/SQL terminado correctamente.
Escenario 2: Error de integridad referencial capturado (idAfiliado=999, nit=1): ORA-02291: restricción de integridad (ISIS2304D25202510.SYS_C001439345) violada - clave principal no encontrada

Procedimiento PL/SQL terminado correctamente.
Escenario 3: Error de restricción de chequeo capturado: ORA-02290: restricción de control (ISIS2304D25202510.SYS_C001439278) violada

Procedimiento PL/SQL terminado correctamente.
```

En el Escenario 1 se verifica la restricción de unicidad de la clave primaria en la tabla IPSs. Se intenta insertar dos registros con el mismo valor de la clave primaria (nit = 100). La primera inserción se realiza con éxito, lo que demuestra que el sistema acepta un valor único asignado manualmente. Sin embargo, al intentar insertar una segunda IPS con el mismo nit, se genera un error ORA-00001, lo que confirma que la restricción de unicidad se está aplicando correctamente. Este comportamiento garantiza que no se puedan duplicar los identificadores de las IPS, asegurando la integridad de la información.

El Escenario 2 evalúa la integridad referencial mediante claves foráneas en la tabla Afiliado_IPS. Se realiza primero una inserción con valores válidos, asumiendo que el afiliado con id 1 y la IPS con nit 1 existen en el sistema; esta operación se completa exitosamente, lo

que confirma que la relación entre tablas está bien definida. Luego, se intenta insertar un registro utilizando un idAfiliado inexistente (999) y un nit válido, lo que provoca un error ORA-02291 (clave principal no encontrada). Este resultado demuestra que la base de datos impide la inserción de datos que violen la integridad referencial, ya que el valor de la clave foránea debe existir en la tabla referenciada.

En el Escenario 3 se prueba el cumplimiento de las restricciones de chequeo definidas en la tabla Afiliados. Se intenta insertar un registro en el que el campo tipoAfiliado tiene un valor inválido ('INVALIDO') que no corresponde a ninguno de los valores permitidos ('CONTRIBUYENTE' o 'BENEFICIARIO'). La operación genera un error ORA-02290, lo que confirma que la restricción de chequeo está funcionando correctamente y evita la inserción de datos que no cumplan con las reglas predefinidas. Este control es fundamental para asegurar que la información almacenada en la base de datos se ajuste a los criterios establecidos en el modelo.

Los resultados obtenidos en la consola, junto con los mensajes de error, confirman que el sistema de base de datos está aplicando correctamente las restricciones de unicidad, integridad referencial y chequeo de valores, garantizando la consistencia y la validez de los datos en cada operación. Estos resultados se ilustran en la imagen adjunta, donde se puede observar el mensaje de error capturado en cada uno de los escenarios de prueba.

El usuario con el cual se trabajo es: ISIS2304D12202510