Herramientas de Programación en Bioinformática y Biología Computacional

Proyecto de Base de Datos de Pruebas Hospitalarias

Autoras:

Deyanira Borroto Alburquerque
Tania Gonzalo Santana

Noviembre 2024

Universidad Autónoma de Madrid

Índice

1. Introducción al problema a resolver

1.1. Contexto

En el ámbito de la salud, la gestión eficiente y precisa de la información es crucial para garantizar la calidad de la atención médica. Los hospitales y centros de salud realizan una gran cantidad de pruebas diagnósticas diariamente, generando un volumen significativo de datos difíciles de manejar que deben ser gestionados de manera efectiva. Esta información relacionada con las pruebas hospitalarias incluye datos de pacientes, médicos, pruebas, muestras obtenidas, resultados y facturación, entre otros.

La digitalización de los registros médicos ha transformado la manera en que se maneja toda esta información en el sector salud. Sin embargo, muchos sistemas aún enfrentan desafíos significativos en términos de integración, accesibilidad y precisión de los datos. Esto puede llevar a errores en el diagnóstico y retrasos en el tratamiento, afectando esto a la salud y supervivencia de los pacientes.

El diseño y desarrollo de una base de datos relacional para la gestión de pruebas hospitalarias ofrece múltiples beneficios: mejora en la precisión de los datos, eficiencia operativa, seguridad y privacidad de los datos de los pacientes y facilidad de acceso, entre otras. Por ejemplo, una base de datos relacional de este tipo permitiría reducir los errores humanos al automatizar la entrada y manejo de datos, asegurando que información crítica como pueden ser resultados de las pruebas y los datos de los pacientes sea precisa y actualizada.

Puesto que la implementación de una base de datos relacional para la gestión de pruebas hospitalarias no solo optimiza la administración de la información, sino que también contribuye a mejorar la calidad de la atención médica, la seguridad del paciente y la eficiencia operativa del centro de salud. Este proyecto busca desarrollar un sistema que aborde estos desafíos y proporcione una solución integral para la gestión de pruebas hospitalarias.

1.2. Objetivo

El objetivo principal de este proyecto es diseñar y desarrollar un modelo de base de datos relacional para gestionar los registros de pruebas hospitalarias. Esto incluye organizar y almacenar información de pacientes, médicos, pruebas, muestras, resultados y facturas de una manera que garantice la integridad de los datos y una recuperación eficiente.

1.3. Alcance

Este proyecto cubrirá la gestión de pacientes, médicos, pruebas, muestras, resultados y facturas, asegurando que todos los datos estén correctamente estructurados y relacionados.

2. Diagrama de Entidad-Relación (ER)

Una vez identificado el problema a resolver, debemos modelarlo mediante un diagrama Entidad-Relación (ER). A continuación se indican los pasos a seguidos para la obtención del digrama ER de nuestra base de datos a la se le ha denominado "Base de Datos de Pruebas Hospitalarias".

2.1. Identificación de entidades y atributos

El primer paso en la creación del diagrama ER es identificar las entidades, los atributos y las relaciones entre ellas. En este caso, hemos definido seis entidades: paciente, médico, muestra, prueba, resultado y factura. Cada una de estas entidades tiene una serie de atributos, es decir, características o propiedades que describen a cada entidad.

A continuación se muestran las entidades con sus respectivos atributos y una breve descripción de los mismos:

Paciente

- **ID_paciente**: Identificador único del paciente.
- **nombre**: Nombre del paciente.
- **apellido**: Apellido del paciente.
- **DNI**: Documento Nacional de Identidad del paciente.
- **telefono**: Número de teléfono del paciente.
- **email**: Correo electrónico del paciente.
- **direccion**: Dirección del paciente.
- **fecha_nacimiento**: Fecha de nacimiento del paciente.
- **sexo**: Sexo del paciente.
- **numero_SS**: Número de Seguridad Social del paciente.

Médico

- **ID_medico**: Identificador único del médico.
- **nombre**: Nombre del médico.
- **apellido**: Apellido del médico.
- **no_colegiado**: Número de colegiado del médico.
- **telefono**: Número de teléfono del médico.
- **email**: Correo electrónico del médico.
- **especialidad**: Especialidad del médico.

Muestra

- **ID_muestra**: Identificador único de la muestra.
- **tipo_muestra**: Tipo de muestra recogida.
- **fecha_obtencion**: Fecha en la que se obtuvo la prueba.
- **condiciones_almacenamiento**: Condiciones de almacenamiento de la prueba.
- **sitio_recogida**: Sitio donde se recogió la muestra para la prueba.

Prueba

- **ID_prueba**: Identificador único de la prueba.
- **tipo_prueba**: Tipo de prueba realizada.
- **Departamento**: Departamento a cargo de la prueba.
- **tipo_muestra**: Tipo de muestra recogida.

Resultado

- **ID_resultado**: Identificador único del resultado.
- **valor_resultado**: Valor del resultado obtenido.
- **valor_normal**: Valor normal de referencia para el resultado.
- **fecha_emision**: Fecha de emisión del resultado.
- **emisor_responsable**: Persona responsable de emitir el resultado.

Factura

- **ID_factura**: Identificador único de la factura.
- **estado_pago**: Estado del pago de la factura (pagado, pendiente, etc.).
- **fecha_emision**: Fecha de emisión de la factura.
- **total_pagar**: Total a pagar en la factura.

2.2. Identificación de relaciones y cardinalidades

La relaciones establecidas entre las diferentes entidades con sus respectivas cardinalidades se recogen en la Tabla 1.

Tabla 1: Relaciones, descripción de las mismas y cardinalidad en la base de datos de Pruebas Hospitalarias

Relación	Descripción	Cardinalidad
Resultado-Prueba	Un resultado corresponde a una prueba	1 a N
Médico-Prueba	Un médico puede solicitar una prueba	1 a N
Muestra - Prueba	Una prueba está asociada a una muestra	1 a N
Paciente - Prueba	Una prueba es realizada a un paciente.	1 a N
Factura - Prueba	Una prueba genera (produce) una factura	1 a N

2.3. Especialización

En nuestro caso hemos optado por un modelo sencillo sin la introducción de especializaciones. No obstante se podrían considerar especializaciones en entidades como Médico (por ejemplo, médicos generales, especialistas) o Prueba (por ejemplo, pruebas de laboratorio, pruebas de imagen). Esto permitiría hacer relaciones como que un médico general no puede pedir pruebas de imagen mientras que un médico especializado si.

2.4. Diagrama ER

Tras la identificación de las entidades, atributos, relaciones y cardinalidades podemos plantear el diagrama ER para nuestra base de datos de Pruebas Hospitalarias. Este se recoge en la Figura 1.

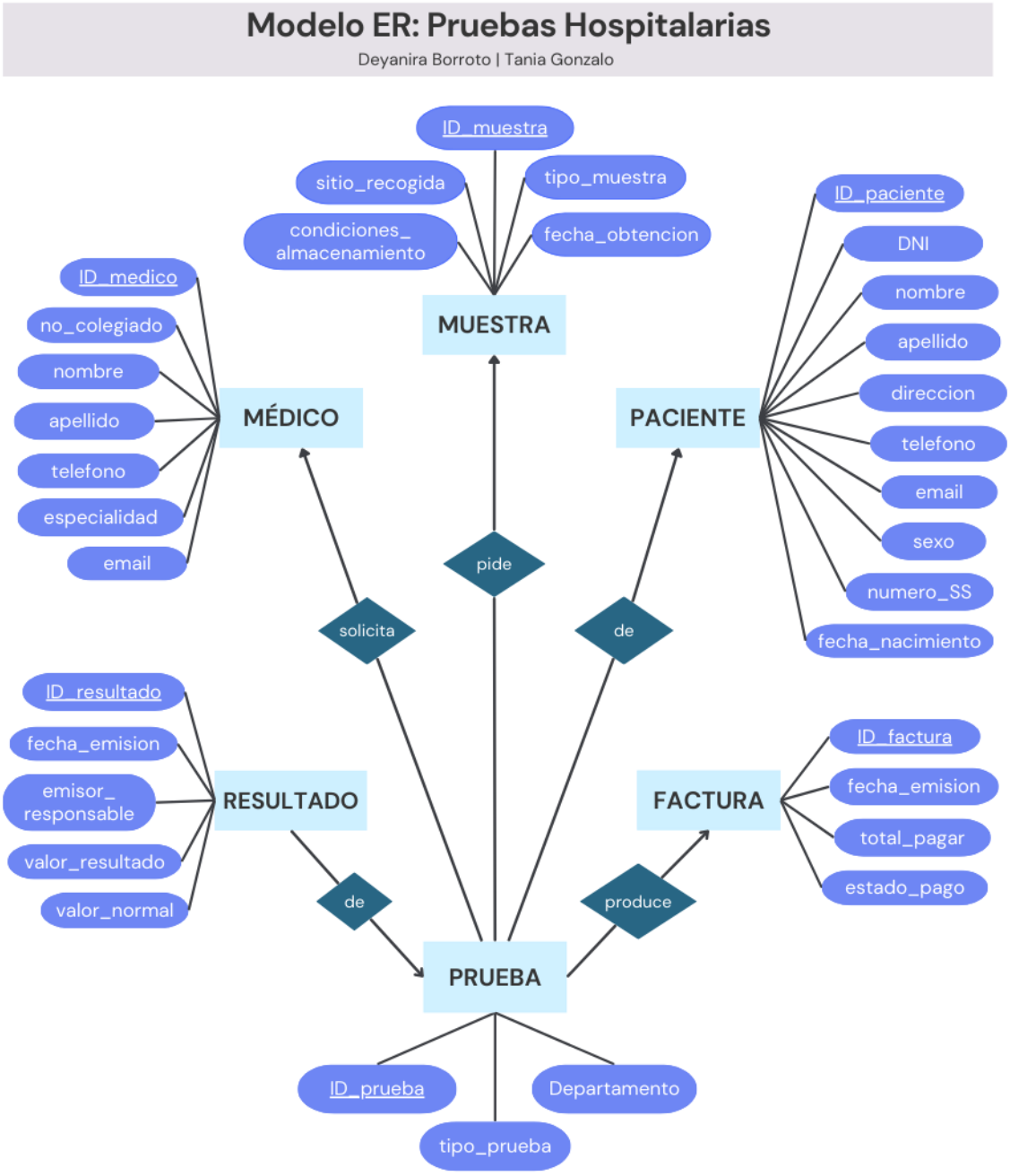

Figura 1: Diagrama ER de la base de datos de Pruebas Hospitalarias. *Hecho en canva.com*

3. Modelo relacional

Una vez establecido el modelo conceptual (diagrama ER) para nuestra base de datos de Pruebas Hospitalarias, debemos transformarlo en un modelo lógico (tablas relacionales), el cual pueda ser manejado por los sistemas de gestión de bases de datos relacionales (SGBDR).

Esto es posible gracias a seguir unas reglas establecidas. Los pasos seguidos en nuestro caso se muestran en las seguientes secciones.

3.1. Identificar claves primarias, entidades y relaciones N-N

En la Tabla 2 se muestra un resumen de las entidades de nuestra base de datos junto con sus respectivas claves primarias. En nuestro caso, no tenemos relaciones N-N.

Tabla 2: Entidades y sus claves primeras correpondientes

Entidad	Clave Primaria
Paciente	ID_paciente
Médico	ID_medico
Prueba	ID_prueba
Muestra	ID_muestra
Resultado	ID_resultado
Factura	ID_factura

Cada una de las entidades del diagrama ER se transformará en una tabla en el modelo relacional. Todas las relaciones en nuestro esquema son de tipo 1-N (uno a muchos). Este tipo de relaciones no se representan en forma de tabla como en el caso de las relaciones N-N, sino que se representan añadiendo una columna a la tabla correspondiente a la entidad del lado "muchos", la cual actuará como una clave foránea haciendo referencia a la clave primaria de la entidad del lado üno".

3.2. Modelo relacional

Tras la identificaciones de las entidades con sus claves primeras y considernado las relaciones 1-N, obtener el modelo relacional de nuestra base de datos de Pruebas Hospitalarias es trivial. Se muestra a continuación:

- **Paciente** (ID_paciente, nombre, apellido, DNI, dirección, telefono, email, fecha_nacimiento, sexo, no_seguridad_social)
- **Médico** (ID_medico, nombre, apellido, no_colegiado, telefono, email, especialidad)
- **Prueba** (ID_prueba, ID_factura ↑, ID_paciente ↑, ID_muestra ↑, ID_medico ↑, tipo_prueba, departamento)
- **Muestra** (ID_muestra, fecha_obtencion, sitio_recogida, tipo_muestra, condiciones_almacenamiento)
- **Resultado** (ID_resultado, ID_prueba ↑, fecha_emisión, emisor_responsable, valor_normal, valor_resultado)
- **Factura** (ID_factura, fecha_emision, total_pagar, estado_pago)

Además, en la Figura 2 muestra el esquema tabular de nuestro modelo.

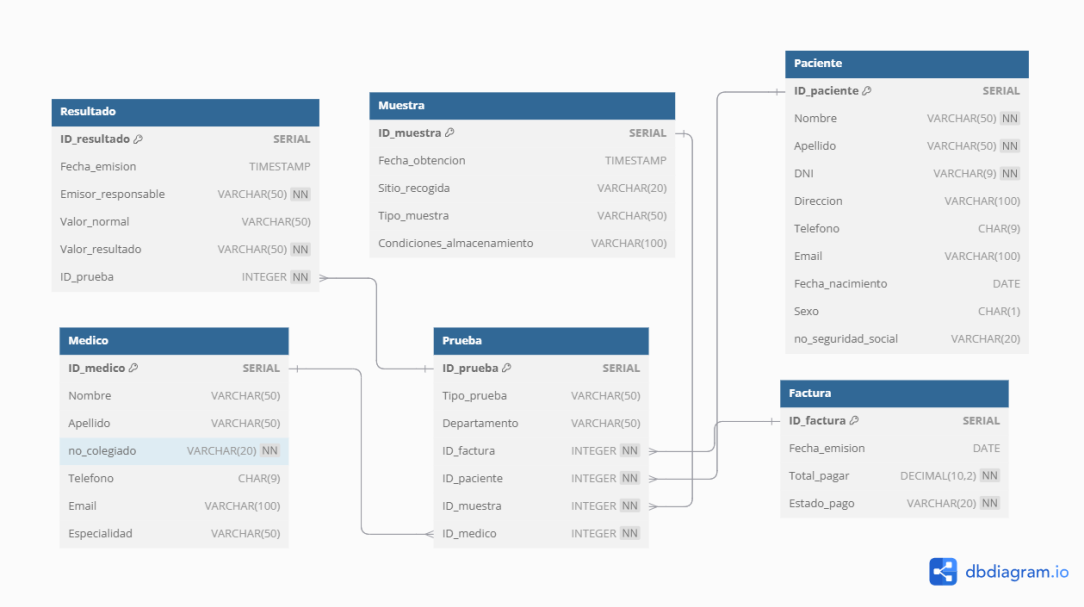

Figura 2: Esquema tabular del diagrama ER de la base de datos de Pruebas Hospitalarias. *Hecho en dbdiagram.io*

4. Creación de tablas e inserción de datos

Una vez obtenido el modelo de relaciones, la creación de tablas e inserción de datos se vuelve trivial gracias al lenguaje SQL. En nuestro caso, utilizamos PostgreSQL como sistema gestor de bases de datos (SGBD). El código SQL utilizado para la creación de las tablas se encuentra en el archivo `CREATE_TABLES.sql`, mientras que los datos insertados están en el archivo `INSERTS_TABLES.sql`.

A continuación, se describe la creación de cada una de las tablas y se muestra el resultado tras la inserción de los datos.

4.1. Tabla Paciente

La **tabla Paciente** almacena información sobre los pacientes, incluyendo su `ID_paciente`, que es de tipo `SERIAL` y además actúa como clave primaria de la entidad. El resto de los atributos son: `Nombre` y `Apellido`, de tipo `VARCHAR(50)`, no pudiendo ser nulos; `DNI`, de tipo `VARCHAR(9)`, que no puede ser nulo y debe ser único; `Direccion` y `Email`, de tipo `VARCHAR(100)`; `Telefono`, de tipo `VARCHAR(9)`; `Fecha_nacimiento`, de tipo `DATE`; `Sexo`, de tipo `CHAR(1)`; y `no_seguridad_social`, de tipo `VARCHAR(20)`. Además, mediante la restricción `CHECK`, se asegura que al menos uno de los campos de contacto (`Telefono` o `Email`) esté presente.

En esta tabla **Paciente** se realizaron 20 inserciones (archivo `INSERTS_TABLES.sql`), obteniendo finalmente el resultado que se muestra en la Tabla 3.

id_ paciente	nombre	apellido	dni	direccion	telefono	email	fecha_ nacimiento	sexo	no_seguridad _social
1	Juan	Perez	12113678A	Calle Alegria	555123456	NULL	1960-01-01	M	123444789
2	Maria	Gamez	87694521B	Avenida Plaza Central	NULL	maria@gmail.com	1946-05-05	F	NULL
3	Marta	Maria	87888881A	Calle Fuencarral	NULL	m.maria@gmail.com	1956-05-05	F	155544789
4	Maria	Peres	87421471W	Calle America	656856789	m.perez@educa.es	1998-04-12	F	155544789
5	Lucia	Fernandez	23731789C	Calle Real 145	555123789	lucia@estudiante.uam.es	1985-03-15	F	987653841
6	Miguel	Sanchez	34426890D	Calle Mayor 101	NULL	miguel_san@gmail.com	1950-07-20	M	NULL
7	Carlos	Ramirez	12389678Z	Calle Sol 15	555987123	NULL	1982-06-15	M	ABC123456
8	Ana	Santos	87654321X	Avenida Luna 5	NULL	ana.santos@gmail.com	1990-02-20	F	DEF987654
9	Pedro	Mendez	23411189Y	Calle Estrella 9	555123654	NULL	1975-10-10	M	GHI456789
10	Laura	Martinez	98456632W	Plaza Mayor 2	NULL	laura.martinez@gmail.com	1988-08-25	F	NULL
11	Manuel	Vega	34788890V	Calle Lluvia 33	555456987	manuel.vega@hotmail.com	1960-04-18	M	JKL321987
12	Miguel	Vega	54322278P	Calle Fuego 42	555789321	NULL	1992-12-12	M	MNO654123
13	Ana	Saez	11189611A	Calle Uno	444489321	NULL	1981-03-05	F	111000111
14	Rebeca	Martin	11777112A	Plaza Mayor	111789321	rebe_martin@hotmail.com	1981-01-01	F	NULL
15	Luis	Rodriguez	87125521B	Calle Alegria	554569321	NULL	1981-01-01	M	111258113
16	Ines	Gill	11111114A	Calle Sonrisa	555896321	NULL	1988-01-01	F	111251114
17	John	Dow	23697189Y	Calle Sorolla	111111321	NULL	1997-09-23	M	987653555
18	Laura	Perez	23999189Y	Calle Pintor Rosales	125896521	lau_perez@hotmail.com	1981-12-15	F	987655555
19	Juan	Terez	99911189Y	Calle Federico Garcia Lorca	125659321	juan_terez@hotmail.com	1985-01-20	M	987611111
20	Lorenzo	Garcia	23331189Y	Calle Severo Ochoa	789985321	NULL	1985-11-20	F	NULL

Tabla 3: Contenido de la Tabla Paciente tras la inserción de datos.

4.2. Tabla Médico

La **tabla Medico** permite almacenar información sobre los médicos, incluyendo su `ID_medico`, que es de tipo `SERIAL` y actúa como clave primaria de la entidad.El resto de atributos son: `Nombre` y `Apellido`, de tipo `VARCHAR(50)`; `no_colegiado`, de tipo `VARCHAR(20)`, que no puede ser nulo y debe ser único; `Telefono`, de tipo `CHAR(9)`; `Email`, de tipo `VARCHAR(100)`; y `Especialidad`, de tipo `VARCHAR(50)`.

En esta tabla **Medico** se realizaron 20 inserciones (archivo `INSERTS_TABLES.sql`), obteniendo finalmente el resultado que se muestra en la Tabla 4.

id_medico	nombre	apellido	no_colegiado	telefono	email	especialidad
1	Dr. Carlos	Lopez	COL003	555654321	carlos@centrosalud.com	Cardiologia
2	Dra. Ana	Martinez	COL406	555987654	ana@hotmail.com	Hematologia
3	Dr. Pedro	Garcia	COL701	555321654	pedro@gmail.com	Dermatologia
4	Dra. Laura	Hernandez	COL012	555654987	laura@gmail.com	Inmunologia
5	Dr. Juan	Ramirez	COL004	555123456	juan.ramirez@hospital.com	Cardiologia
6	Dr. Alberto	Santos	COL543	555654321	alberto.santos@hospital.com	Oncologia
7	Dra. Juana	Lopez	COL987	955789123	juana.lopez@hospital.com	Dermatologia
8	Dra. Alejandra	Martinez	COL246	055321654	alejandra.martinez@gmail.com	Neurologia
9	Dra. Laura	Gonzalez	COL135	545987456	laura.gonzalez@hotmail.com	Med.Interna
10	Dr. Miguel	Ruiz	COL864	575654987	miguel.ruiz@hospital.com	Oncologia
11	Dr. Tio	Guay	COL111	218956111	tioguay@gmail.com	Hematologia
12	Dra. Lucy	Delight	COL112	221611481	lucy@gmail.com	Cardiologia
13	Dra. Pepi	Santana	COL113	231156111	pepi@gmail.com	Med.General
14	Dr. Keith	Sunny	COL114	241459611	NULL	Hematologia
15	Dr. John	Galea	COL115	251555611	john@gmail.com	Med.General
16	Dra. Jul	Step	COL116	261145661	jul@gmail.com	Med.General
17	Dra. Sand	Likelihood	COL117	271259811	sand@gmail.com	Traumatologia
18	Dr. Arbol	Lively	COL118	281658911	arbol@gmail.com	Endocrinologia
19	Dr. Trevor	Noah	COL119	291125411	noah@gmail.com	Neumologia
20	Dra. Sandra	Pepi	COL120	212125911	sandra@gmail.com	Endocrinologia

Tabla 4: Contenido de la Tabla Médico tras la inserción de datos.

4.3. Tabla Prueba

La **tabla Prueba** incluye información sobre las pruebas realizadas y permite la relación con los pacientes, médicos, muestras y facturas correspondientes. Como atributos tenemos su `ID_prueba`, que es de tipo `SERIAL` y actúa como clave primaria de la entidad. Los demás atributos son: `Tipo_prueba`, de tipo `VARCHAR(50)`; `Departamento`, de tipo `VARCHAR(50)`; `ID_factura`, de tipo `INTEGER`, que referencia a `Factura(ID_factura)` y no puede ser nulo; `ID_paciente`, de tipo `INTEGER`, que referencia a `Paciente(ID_paciente)` y no puede ser nulo; `ID_muestra`, de tipo `INTEGER`, que referencia a `Muestra(ID_muestra)` y no puede ser nulo; y `ID_medico`, de tipo `INTEGER`, que referencia a `Medico(ID_medico)` y no puede ser nulo.

En esta tabla **Prueba** se realizaron 28 inserciones (archivo `INSERTS_TABLES.sql`), obteniendo finalmente el resultado que se muestra en la Tabla 5.

id_prueba	tipo_prueba	departamento	id_factura	id_paciente	id_muestra	id_medico
1	Electrocardiograma	Cardiologia	1	1	1	1
2	Coagulacion	Bioquimica	2	2	2	2
3	Biopsia de piel	Patologia	3	3	3	3
4	Serologia	Bioquimica	4	4	4	4
5	Analitica	Hematologia	5	5	5	5
6	Holter	Cardiologia	6	6	6	5
7	Biopsia	Patologia	7	7	7	6
8	Electrocardiograma	Cardiologia	8	8	8	5
9	Prueba de esfuerzo	Deporte	9	9	9	9
10	Biopsia	Patologia	10	10	10	10
11	Analitica	Hematologia	11	11	11	11
12	Holter	Cardiologia	11	11	23	12
13	Analitica	Hematologia	12	12	12	12
14	Biopsia	Patologia	13	13	13	13
15	Analitica	Hematologia	1	1	1	13
16	Hematologia	Hematologia	14	14	14	14
17	Coagulacion	Bioquimica	1	1	1	14
18	Serologia	Bioquimica	2	2	2	14
19	Analitica	Hematologia	15	15	15	15
20	Microbiología	Microbiología	15	15	15	15
21	Biopsia	Patologia	16	16	16	16
22	Rayos X	Radiologia	17	17	17	17
23	Hormonas	Endocrinologia	18	18	18	18
24	Espirometria	Neumologia	19	19	19	19
25	Ecografia de cuello	Radiologia	20	20	24	20
26	Analitica	Hematologia	20	20	20	1
27	Biopsia de piel	Patologia	1	1	21	7
28	Escaner	Radiologia	7	7	22	8

Tabla 5: Contenido de la Tabla Prueba tras la inserción de datos.

4.4. Tabla Muestra

La **tabla Muestra** permite almacenar información sobre las muestras recogidas, incluyendo su `ID_muestra`, que es de tipo `SERIAL` y actúa como clave primaria de la entidad. El resto de atributos son: `Fecha_obtencion`, de tipo `TIMESTAMP`; `Sitio_recogida`, de tipo `VARCHAR(20)`; `Tipo_muestra`, de tipo `VARCHAR(50)`; y `Condiciones_almacenamiento`, de tipo `VARCHAR(100)`, pudiendo todos estos atributos (menos la clave primaria) ser nulos.

En esta tabla **Muestra** se realizaron 24 inserciones (archivo `INSERTS_TABLES.sql`), obteniendo finalmente el resultado que se muestra en la Tabla 6.

id_muestra	fecha_obtencion	sitio_recogida	tipo_muestra	condiciones_almacenamiento
1	2024-06-01 08:00:00	Extracciones	Sangre	Refrigerado
2	2024-06-01 08:00:00	Extracciones	Sangre	Refrigerado
3	2024-06-02 08:00:00	Quirofano1	Tejido	Congelado
4	2024-06-04 08:00:00	Extracciones	Sangre	Congelado
5	2024-06-05 08:00:00	Laboratorio Central	Sangre	Temperatura ambiente
6	NULL	NULL	NULL	NULL
7	2024-06-05 08:00:00	Quirofano1	Tejido	Congelado
8	NULL	NULL	NULL	NULL
9	NULL	NULL	NULL	NULL
10	2024-06-06 08:00:00	Quirofano3	Tejido	Congelado
11	2024-06-07 08:30:00	Extracciones	Sangre	Temperatura ambiente
12	2024-06-08 08:20:00	Extracciones	Sangre	Temperatura ambiente
13	2024-06-10 08:16:00	Quirofano4	Tejido	Congelado
14	2024-06-11 08:16:00	Extracciones	Sangre	Temperatura ambiente
15	2024-06-21 07:00:00	Extracciones	Sangre	Temperatura ambiente
16	2024-06-22 09:00:00	Quirofano2	Tejido	Congelado
17	NULL	NULL	NULL	NULL
18	2024-06-12 10:00:00	Extracciones	Sangre	Refrigerado
19	NULL	NULL	NULL	NULL
20	2024-06-13 18:00:00	Extracciones	Sangre	Temperatura ambiente
21	2024-06-14 20:00:00	Quirofano1	Tejido	Congelado
22	NULL	NULL	NULL	NULL
23	NULL	NULL	NULL	NULL
24	NULL	NULL	NULL	NULL

Tabla 6: Contenido de la Tabla Muestra tras la inserción de datos.

4.5. Tabla Factura

La **tabla Factura** permite almacenar información sobre las facturas emitidas. Incluye su `ID_factura`, que es de tipo `SERIAL` y actúa como clave primaria de la entidad. El resto de atributos son: `Fecha_emision`, de tipo `DATE`; `Total_pagar`, de tipo `DECIMAL(10, 2)`, que no puede ser nulo; y `Estado_pago`, de tipo `VARCHAR(20)`, que tampoco puede ser nulo.

En esta tabla **Factura** se realizaron 20 inserciones (archivo `INSERTS_TABLES.sql`), obteniendo finalmente el resultado que se muestra en la Tabla 7.

id_factura	fecha_emision	total_pagar	estado_pago
1	2024-09-20	150.75	Pagado
2	2024-09-21	200.00	Pendiente
3	2024-09-24	75.50	Pagado
4	2024-09-25	120.00	Pendiente
5	2024-09-15	50.75	Pagado
6	2024-09-20	100.00	Pendiente
7	2024-09-05	175.50	Pagado
8	2024-09-12	220.00	Pendiente
9	2024-10-10	300.25	Pagado
10	2024-10-01	150.00	Pendiente
11	2024-10-04	400.00	Pendiente
12	2024-10-16	300.00	Pagado
13	2024-10-18	200.00	Pagado
14	2024-10-20	410.99	Pagado
15	2024-10-01	340.99	Pendiente
16	2024-10-01	400.00	Pendiente
17	2024-10-01	110.00	Pendiente
18	2024-10-01	400.00	Pagado
19	2024-10-01	210.00	Pendiente
20	2024-10-01	520.00	Pagado

Tabla 7: Contenido de la Tabla Factura tras la inserción de datos.

4.6. Tabla Resultado

La **tabla Resultado** almacena los resultados de las pruebas. Incluye su `ID_resultado`, que es de tipo `SERIAL` y actúa como clave primaria de la entidad. Los demás atributos son: `Fecha_emision`, de tipo `TIMESTAMP`; `Emisor_responsable`, de tipo `VARCHAR(50)`, que no puede ser nulo; `Valor_normal`, de tipo `VARCHAR(50)`; `Valor_resultado`, de tipo `VARCHAR(50)`, que no puede ser nulo; y `ID_prueba`, de tipo `INTEGER`, que referencia a `Prueba(ID_prueba)` y no puede ser nulo.

En esta tabla **Resultado** se realizaron 35 inserciones (archivo `INSERTS_TABLES.sql`), obteniendo finalmente el resultado que se muestra en la Tabla 8.

id_ resultado	fecha_emision	emisor_responsable	valor_normal	valor_resultado	id_ prueba
1	2024-07-01 10:00:00	Dra. Raquel Aguado	Rango Normal	Rango Normal	1
2	2024-07-02 10:00:00	Dr. José Izquierdo	Negativo	Negativo	2
3	2024-07-20 10:00:00	Dr. José Izquierdo	Negativo	Positivo	2
4	2024-07-21 10:00:00	Dra. Eva Iglesias	Negativo	Negativo	3
5	2024-07-10 10:00:00	Dra. Ana Segundo	Positivo	Negativo	4
6	2024-08-26 09:00:00	Dra. Ana Segundo	Positivo	Negativo	4
7	2024-07-23 10:00:00	Dra. Nerea Sanchez	Rango Normal	Rango Normal	5
8	2024-07-16 10:00:00	Dr. Carlos Ramirez	Rango Normal	Anomalia	6
9	2024-07-15 10:00:00	Dra. Ana Santos	Normal	Malignidad	7
10	2024-07-13 10:00:00	Dr. Pedro Lopez	Rango Normal	Onda T alterada	8
11	2024-07-12 10:00:00	Dra. Laura Martinez	Normal	Normal	9
12	2024-07-11 10:00:00	Dr. Manuel Gonzalez	Normal	Benigno	10
13	2024-08-18 10:00:00	Dra. Elena Ruiz	Rango Normal	Colesterol alto	11
14	2024-07-20 10:00:00	Dra. Elena Ruiz	Rango Normal	Trigliceridos altos	11
15	2024-07-21 10:00:00	Dra. Elena Ruiz	Rango Normal	Rango Normal	11
16	2024-07-26 10:00:00	Dr. Carlos Lopez	Rango Normal	Rango Normal	12
17	2024-07-30 10:00:00	Dra. Ana Martinez	Rango Normal	Rango Normal	13
18	2024-07-30 10:00:00	Dra. Ana Lopez	Normal	Malignidad	14
19	2024-07-31 10:00:00	Dra. Juana Garcia	Rango Normal	Transaminasas altas	15
20	2024-07-16 10:00:00	Dra. Juana Garcia	Rango Normal	Glucosa baja	15
21	2024-07-05 10:00:00	Dr. Pedro Arroba	Negativo	Negativo	16
22	2024-07-06 10:00:00	Dr. Juan Gonzalez	0.9	0.9	17
23	2024-07-20 10:00:00	Dr. Federico Roldan	Negativo	Positivo	18
24	2024-07-08 10:00:00	Dr. Miguel Galan	Rango Normal	Rango Normal	19
25	2024-07-07 10:00:00	Dr. Miguel Galan	Rango Normal	Celiaquia	19
26	2024-07-16 10:00:00	Dr. Jose Pedro Sanz	Positivo	Negativo	20
27	2024-07-16 10:00:00	Dr. Jose Pedro Sanz	Positivo	Positivo	20
28	2024-07-18 10:00:00	Dra. Maria del Mar Sanchez	Normal	Benigno	21
29	2024-07-10 10:00:00	Dra. Estrella Aguado	Normal	Fractura	22
30	2024-07-22 10:00:00	Dra. Eva Perez	Normal	TSH alta	23
31	2024-07-28 09:32:00	Dra. Sara Gala	Normal	Normal	24
32	2024-07-25 10:32:00	Dr. Manuel Romero	Normal	Nodulos	25
33	2024-07-23 11:26:00	Dr. Oscar Lafuente	Normal	Normal	26
34	2024-07-29 11:26:00	Dr. Juan Perez	Normal	Benigno	27
35	2024-07-28 11:26:00	Dra. Lucia Pau	Normal	Normal	28

Tabla 8: Contenido de la Tabla Resultado tras la inserción de datos.

5. Consultas de ejemplo y resultados

En esta sección se presentan algunas consultas avanzadas que se pueden realizar en nuestra base de datos de pruebas hospitalarias. Estas consultas permiten extraer información útil y compleja, aprovechando las relaciones entre las tablas, poniendo en evidencia la relevancia de las bases de datos en este campo.

El código SQL de cada una de las consultas que se describen a continuación se encuentra en el archivo `QUERIES.sql`.

5.1. Recuperar pacientes con más de una muestra mayores de 50 años.

Consulta SQL

```sql
SELECT Paciente.Nombre AS nombre_paciente, Paciente.Apellido AS apellido_paciente, EXTRACT(YEAR
     FROM AGE(Paciente.Fecha_nacimiento)) AS edad, COUNT(DISTINCT Muestra.ID_muestra) AS
    Numero_Muestras
FROM Paciente
JOIN Prueba ON Paciente.ID_paciente = Prueba.ID_paciente
JOIN Muestra ON Prueba.ID_muestra = Muestra.ID_muestra
WHERE EXTRACT(YEAR FROM AGE(Paciente.Fecha_nacimiento)) > 50
GROUP BY Paciente.Nombre, Paciente.Apellido, Paciente.Fecha_nacimiento
HAVING COUNT(DISTINCT Muestra.ID_muestra) > 1
ORDER BY Numero_Muestras DESC;
```

Explicación de la consulta SQL

Esta consulta recupera el número de muestras distintas asociadas a cada paciente mayor de 50 años. Para ello, se realiza la unión de las tablas **Paciente**, **Prueba** y **Muestra** mediante las claves `ID_paciente` y `ID_muestra`. Posteriormente, se filtran los pacientes mayores de 50 años utilizando la cláusula `WHERE`, que aplica la función `EXTRACT(YEAR FROM AGE(...))` para calcular la edad del paciente a partir de su fecha de nacimiento. Los resultados se agrupan por paciente mediante la cláusula `GROUP BY`, utilizando como identificadores únicos su nombre, apellido y fecha de nacimiento. A continuación, se filtran los grupos para incluir únicamente aquellos pacientes con más de una muestra registrada, utilizando la cláusula `HAVING`. Finalmente, los resultados se ordenan de manera descendente según la cantidad de muestras distintas mediante la cláusula `ORDER BY`, priorizando así los pacientes con más material disponible.

Resultado de la consulta

La salida de la consulta muestra los pacientes mayores de 50 años que tienen más de una muestra registrada en la base de datos como vemos en la Tabla 9.

nombre_paciente	apellido_paciente	edad	numero_muestras
Juan	Perez	64	2
Manuel	Vega	64	2

Tabla 9: Pacientes con más de una muestra mayores de 50 años

Interpretación

La salida de la consulta muestra el nombre y apellido del paciente, su edad y el número de muestras distintas que tienen. En nuestro caso, existen dos pacientes en nuestra base de datos: **Juan Pérez** y **Manuel Vega**, ambos de 64 años, cada uno con 2 muestras distintas recogidas.

Esto indica que en el laboratorio tenemos material adicional disponible para estos pacientes, lo cual podría ser útil en caso de necesitar realizar pruebas complementarias.

5.2. Facturas pendientes de pago con un total a pagar superior a 300

Consulta SQL

```
SELECT
    Paciente.Nombre AS nombre_paciente,
    Paciente.Apellido AS apellido_paciente,
    Factura.Total_pagar AS Pendiente_pago
FROM Paciente
JOIN Prueba ON Paciente.ID_paciente = Prueba.ID_paciente
JOIN Factura ON Prueba.ID_factura = Factura.ID_factura
WHERE Factura.Estado_pago = 'Pendiente' AND Factura.Total_pagar > 300
GROUP BY Paciente.Nombre, Paciente.Apellido, Factura.Total_pagar
ORDER BY Pendiente_pago DESC;
```

Explicación de la consulta SQL

Esta consulta recupera el nombre y apellido de los pacientes, junto con el total pendiente de pago de las facturas, ordenados de manera descendente según el total a pagar pendiente. Para ello, se realiza la unión de las tablas **Paciente**, **Prueba** y **Factura** mediante las claves `ID_paciente` e `ID_factura`. La cláusula `WHERE` filtra los resultados para incluir solo los pacientes que deben pagar más de 300 (`Total_pagar >300`) y las facturas cuyo estado de pago es `'Pendiente'`. Los resultados se agrupan por paciente y total pendiente de pago utilizando la cláusula `GROUP BY`. Finalmente se ordenan según el total a pagar pendiente mediante la cláusula `ORDER BY`.

Resultado de la consulta

Como podemos observar en la Tabla 10, la consulta devuelve los nombres y apellidos de los pacientes que tienen facturas pendientes de pago con un total a pagar superior a 300.00.

nombre_paciente	apellido_paciente	pendiente_pago
Ines	Gill	400.00
Manuel	Vega	400.00
Luis	Rodriguez	340.99

Tabla 10: Pacientes con facturas pendientes de pago mayores a 300.00.

Interpretación Los resultados muestran a los pacientes con su respectiva cantidad de pagar pendiente. En nuestro caso, vemos que solo tenemos tres pacientes, de los cuales **Ines Gill** y **Manuel Vega** tiene pendiente pagar un total de 400, mientras que **Luis Rodriguez** tiene utiene pendiente pagar un total de 340.99.

Esta información facilita la gestión de cuentas por cobrar con valores elevados (superiores a 300) que todavía no han sido pagadas.

5.3. Obtener el historial de pruebas y resultados de un paciente específico.

Consulta SQL

```
SELECT
    p.Nombre, p.Apellido, pr.Tipo_prueba, r.Fecha_emision, r.Valor_resultado
FROM Paciente p
JOIN Prueba pr ON p.ID_paciente = pr.ID_paciente
JOIN Resultado r ON pr.ID_prueba = r.ID_prueba
WHERE p.DNI = '12113678A'
ORDER BY r.Fecha_emision DESC;
```

Explicación de la consulta SQL

La consulta permite recuperar el historial de pruebas y resultados de un paciente específico identificado por su DNI. Para ello, se realiza la unión de las tablas **Paciente**, **Prueba** y **Resultado** mediante las claves `ID_paciente` y `ID_prueba`. Se utiliza la cláusula `JOIN` para combinar las tablas y la cláusula `WHERE` para filtrar los resultados basados en el DNI del paciente. La cláusula `SELECT` permite elegir las columnas relevantes, como el nombre y apellido del paciente, el tipo de prueba, la fecha de emisión y el valor del resultado. Además, se ordena por fecha de emision del resultado de la prueba mediante `ORDER BY` de más reciente a más antigua.

Resultado de la consulta

La salida de la consulta muestra el historial de pruebas y resultados del paciente Juan Perez, con el DNI `'12113678A'`. Los resultados incluyen el tipo de prueba, la fecha de emisión y el valor del resultado de cada prueba realizada como podemos observar en la Tabla 11.

nombre	apellido	tipo_prueba	fecha_emision	valor_resultado
Juan	Perez	Analítica	2024-07-31 10:00:00	Transaminasas altas
Juan	Perez	Biopsia de piel	2024-07-29 11:26:00	Benigno
Juan	Perez	Analítica	2024-07-16 10:00:00	Glucosa baja
Juan	Perez	Coagulación	2024-07-06 10:00:00	0.9
Juan	Perez	Electrocardiograma	2024-07-01 10:00:00	Rango Normal

Tabla 11: Historial de pruebas y resultados del paciente con DNI '12113678A'

Interpretación

En este caso podemos observar el historial de pruebas y resultados del paciente **Juan Perez**, con DNI `'12113678A'`. La tabla muestra las pruebas realizadas a este paciente como por ejemplo un electrocardiograma con resultado de `Rango Normal`, y una analítica con resultados de `Transaminasas altas`, entre otros.

Esta información podría ser crucial para el seguimiento de la salud del paciente y para los médicos al revisar los resultados anteriores y tomar decisiones informadas.

5.4. Importe de las pruebas solicitadas por cada uno de los médicos en la segunda quincena de septiembre de 2024 ordenado por dia.

Consulta SQL

```
SELECT m.Nombre AS nombre_medico,
       TO_CHAR(f.Fecha_emision, 'DD') AS Dia,
       f.Total_pagar AS importe_por_pruebas_solicitadas
FROM Factura f
JOIN Prueba p ON p.id_factura = f.id_factura
JOIN Medico m ON p.id_medico = m.ID_medico
WHERE f.Fecha_emision BETWEEN '2024-09-15' AND '2024-09-30'
GROUP BY m.Nombre, TO_CHAR(f.Fecha_emision, 'DD'), f.Total_pagar
ORDER BY Dia ASC;
```

Explicación de la consulta SQL

La consulta obtiene el nombre de los médicos, el día del mes de la fecha de emisión y el importe por las pruebas solicitadas en un período específico. Se realiza la unión de las tablas **Factura**, **Prueba** y **Medico** mediante las claves `id_factura` y `id_medico`. La cláusula `JOIN` se utiliza para combinar las tablas y `SELECT` para elegir las columnas relevantes, como el nombre del médico, el día de la emisión y el importe total de la factura. La cláusula `WHERE` filtra los resultados para incluir solo aquellas facturas emitidas entre el 15 y el 30 de septiembre de 2024. Los resultados se agrupan por el nombre del médico,

el día de la emisión y el importe, utilizando la cláusula `GROUP BY`, y se ordenan por día de manera ascendente con la cláusula `ORDER BY`.

Resultado de la consulta

La salida de la consulta (Tabla 12) muestra los nombres de los médicos, el día de la fecha de emisión y el importe por las pruebas solicitadas dentro del período comprendido entre el 15 y el 30 de septiembre de 2024.

nombre_medico	dia	importe_por_pruebas_solicitadas
Dr. Juan	15	50.75
Dr. Carlos	20	150.75
Dr. Juan	20	100.00
Dr. Keith	20	200.00
Dra. Juana	20	150.75
Dra. Pepi	20	150.75
Dr. Keith	21	200.00
Dra. Ana	21	200.00
Dr. Pedro	24	75.50
Dra. Laura	30	120.00

Tabla 12: Importe diario de pruebas solicitadas por médicos en la segunda quincena de septiembre de 2024

Interpretación

En este caso, podemos observar el dia 15 de septiembre se emitió una factura asociada al `Dr. Juan` asociada a un importe total a pagar de 50.75, mientras que el 30 de septiembre se emitió una factura asociada a la `Dra. Laura` asociada a un importe total a pagar de 120.00.

Esta consulta permite obtener el importe por pruebas solicitadas asociadas a cada uno de los médicos por dia, pudiendo esto ser de utilidad para la gestión de gastos del hospital.

5.5. El paciente con más pruebas realizadas a partir de una única muestra que debe mantenerse o bien refrigeradas o bien congeladas.

Consulta SQL

```
SELECT p.nombre AS nombre_paciente,
       m.Tipo_muestra,
       m.Condiciones_almacenamiento,
       COUNT(pr.ID_prueba) AS Numero_Pruebas
FROM Muestra m
JOIN Prueba pr ON m.ID_muestra = pr.ID_muestra
JOIN Paciente p ON p.ID_paciente = pr.ID_paciente
WHERE LOWER(m.Condiciones_almacenamiento) LIKE '%refrigerado%'
   OR LOWER(m.Condiciones_almacenamiento) LIKE '%congelado%'
GROUP BY m.ID_muestra, m.Tipo_muestra, m.Condiciones_almacenamiento, p.nombre
HAVING COUNT(pr.ID_prueba) > 1
ORDER BY Numero_Pruebas DESC
LIMIT 1;
```

Explicación de la consulta SQL

Esta consulta tiene como objetivo encontrar el paciente con el mayor número de pruebas realizadas a partir de una única muestra que debe mantenerse o bien refrigerada o bien congelada. Para ello, seleccionamos varias columnas: el nombre del paciente, el tipo de muestra, las condiciones de almacenamiento y el número de pruebas realizadas. Posteirormente, se usa `JOIN` para combinar las tablas **Muestra**, **Prueba** y **Paciente**. La cláusula `WHERE` permite filtrar las muestras que deben mantenerse o bien refrigeradas

o bien congeladas, y se emplea la función `COUNT` para contar las pruebas asociadas a cada muestra. Luego, los resultados se agrupan por muestra, tipo de muestra, condiciones de almacenamiento y nombre del paciente mediante la cláusula `GROUP BY`, y se filtran para mostrar solo aquellos pacientes con más de una prueba mediante la cláusula `HAVING`. Finalmente, se ordenan los resultados de manera descendente por el número de pruebas realizadas y se limita la salida a un solo registro utilizando `LIMIT 1`.

Resultado de la consulta

Como vemos en la Tabla 13, la salida de la consulta muestra el nombre del paciente, el tipo de muestra, las condiciones de almacenamiento y el número de pruebas realizadas para la muestra que debe mantenerse refrigerada o congelada.

nombre_paciente	tipo_muestra	condiciones_almacenamiento	numero_pruebas
Juan	Sangre	Refrigerado	3

Tabla 13: Pacientes con mayor cantidad de pruebas realizadas a partir de una única muestra que requiere refrigeración o congelación.

Interpretación

En este caso, el paciente con más pruebas realizadas a aprtir de una muestra es `Juan`, quien tiene una muestra de sangre que debe mantenerse refrigerada. El total de pruebas realizadas con esa muestra es de 3.

Esta información es importante para conocer la carga de pruebas asociada a un tipo específico de muestra, lo que puede ser relevante a nivel médico y gestión hospitalaria.

5.6. Obtener el primer médico ordenado por apellido junto con su número de pruebas del departamento con más pruebas solicitadas.

Consulta SQL

```sql
SELECT m.Nombre AS nombre_medico,
      m.Apellido AS apellido_medico,
      COUNT(p.ID_prueba) AS Numero_Pruebas
FROM Medico m
JOIN Prueba p ON m.ID_medico = p.ID_medico
WHERE p.Departamento = (
       SELECT p1.Departamento
       FROM Prueba p1
       GROUP BY p1.Departamento
       ORDER BY COUNT(p1.ID_prueba) DESC
       LIMIT 1)
GROUP BY m.Nombre, m.Apellido
ORDER BY m.Apellido
LIMIT 1;
```

Explicación de la consulta SQL

Esta consulta permite recuperar el nombre, apellido y el número de pruebas realizadas por un médico en el departamento con el mayor número de pruebas. Para ello, se realiza un `JOIN` entre las tablas **Medico** y **Prueba** utilizando el campo `ID_medico`. La consulta filtra los resultados para el departamento con el mayor número de pruebas, lo cual se obtiene mediante una subconsulta que agrupa las pruebas por departamento y ordena los resultados por la cantidad de pruebas realizadas, limitando el resultado a un solo departamento (el que tenga más pruebas). Posteriormente, se agrupan los resultados por el nombre y apellido del médico, se ordenan alfabéticamente por el apellido y se limita la salida a una fila utilizando `LIMIT 1`.

Resultado de la consulta

Como podemos ver en la Tabla 14, la consulta devuelve una tabla con el nombre, apellido y el número de pruebas realizadas por el médico que trabaja en el departamento con el mayor número de pruebas.

nombre_medico	apellido_medico	Numero_Pruebas
Dra. Lucy	Delight	1

Tabla 14: Primer médico por apellido y número de pruebas del departamento con mayor volumen de solicitudes.

Interpretación

Podemos observar que, en este caso, la `Dra. Lucy Delight` es la médico seleccionada. Esto se debe a que, entre todos los médicos que han solicitado pruebas en el departamento con mayor número de pruebas solicitadas (Hematología), el apellido "Delight.[es] el primero en orden alfabético ascendente. Esta médico ha solicitado un total de una prueba en dicho departamento (Hematología).

5.7. Encontrar el promedio de tiempo desde la recogida de muestra hasta la emision de resultado por tipo de muestra

Consulta SQL

```
SELECT
    m.Tipo_muestra,
    FLOOR(AVG(EXTRACT(EPOCH FROM (r.Fecha_emision - m.Fecha_obtencion)) / 86400)) AS
        Promedio_Dias,
    FLOOR(AVG(EXTRACT(EPOCH FROM (r.Fecha_emision - m.Fecha_obtencion)) % 86400) / 60) AS
        Promedio_Minutos
FROM Muestra m
JOIN Prueba pr ON m.ID_muestra = pr.ID_muestra
JOIN Resultado r ON pr.ID_prueba = r.ID_prueba
WHERE m.Tipo_muestra IS NOT NULL
GROUP BY m.Tipo_muestra;
```

Explicación de la consulta SQL

Esta consuta permite calcular el promedio de tiempo desde la recogida de la muestra hasta la emisión de los resultados, agrupado por tipo de muestra. Para ello, se realiza un `JOIN` entre las tablas **Muestra**, **Prueba** y **Resultado**, utilizando los campos `ID_muestra` y `ID_prueba`. Además, se usan las funciones de fecha y hora (`EXTRACT(EPOCH FROM ...)`) para calcular la diferencia entre las fechas de emisión de resultados y la recogida de las muestras, expresada en días y minutos. Finalmente, la consulta agrupa los resultados por el tipo de muestra y calcula el promedio de días y minutos, utilizando `AVG()` para el promedio y `FLOOR()` para redondear los resultados.

Resultado de la consulta

Como podemos observar en la Tabla 15, la consulta nos proporciona el promedio de tiempo en días y minutos desde la recogida de la muestra hasta la emisión de los resultados, agrupado por tipo de muestra.

tipo_muestra	promedio_dias	promedio_minutos
Tejido	40	241
Sangre	41	160

Tabla 15: Promedio de tiempo desde la recogida de muestra hasta la emisión de resultados, desglosado por tipo de muestra.

Interpretación

Podemos observar que, para las muestras de tejido, el promedio es de 40 días y 241 minutos, mientras que para las muestras de sangre, el promedio es de 41 días y 160 minutos. Por lo tanto, ambas tienen un promedio similar. Con esta consulta podemos determinar el *turnaround time* de los resultados para los diferentes tipos de muestras.

5.8. Encontrar los pacientes cuyo resultado más reciente es anormal y se le ha realizado mas de una prueba

Consulta SQL

```
SELECT DISTINCT Paciente.Nombre AS nombre_paciente,
                Paciente.Apellido AS apellido_paciente,
                Resultado.Valor_resultado,
                Resultado.Valor_normal,
                Resultado.Fecha_emision
FROM Paciente
JOIN Prueba ON Paciente.ID_paciente = Prueba.ID_paciente
JOIN Resultado ON Prueba.ID_prueba = Resultado.ID_prueba
WHERE Resultado.Fecha_emision = (
       SELECT MAX(R.Fecha_emision)
       FROM Resultado R
       JOIN Prueba P ON R.ID_prueba = P.ID_prueba
       WHERE P.ID_paciente = Paciente.ID_paciente)
AND Resultado.Valor_resultado <> Resultado.Valor_normal
AND Paciente.ID_paciente IN (
       SELECT ID_paciente
       FROM Prueba
       GROUP BY ID_paciente
       HAVING COUNT(ID_prueba) > 1)
ORDER BY Paciente.Apellido;
```

Explicación de la consulta SQL

Esta consulta permite encontrar los pacientes cuyo resultado más reciente es anormal y que además se les ha realizado más de una prueba. La consulta selecciona gracias al uso de `SELECT` los nombres y apellidos de los pacientes, el valor del resultado, el valor normal y la fecha de emisión del resultado. Mediante `JOIN` se unen las tablas **Paciente**, **Prueba** y **Resultado** mediante las claves `ID_paciente` y `ID_prueba`. Los resultados se filtran según la fecha más reciente de emisión del resultado y que el valor del resultado sea diferente al valor normal mediante el uso de `WHERE`. Además, gracias al uso del operador `AND`, se filtran los pacientes que han tenido más de una prueba realizada utilizando una subconsulta con la función `COUNT()`. Finalmente se ordenan por orden ascendente alfanumérico de los apellidos de los pacientes.

Resultado de la consulta

Como podemos observar en la Tabla 16, la consulta muestra los nombres y apellidos de los pacientes, el valor de su resultado más reciente, el valor normal y la fecha de emisión del resultado, solo para aquellos pacientes que han tenido más de una prueba y cuyo resultado más reciente es anormal.

Interpretación

Podemos observar que el paciente **Juan Pérez** tiene un resultado reciente de *transaminasas altas*, mientras que el valor normal es *Rango Normal*. Además, vemos a la paciente **Maria Gamez** también con resultado anormal, obteniendo un resultado *Positivo* frente a un valor normal *Negativo*.

nombre_paciente	apellido_paciente	valor_resultado	valor_normal	fecha_emision
Maria	Gamez	Positivo	Negativo	2024-07-20 10:00:00
Lorenzo	Garcia	Nodulos	Normal	2024-07-25 10:32:00
Juan	Perez	Transaminasas altas	Rango Normal	2024-07-31 10:00:00
Luis	Rodriguez	Negativo	Positivo	2024-07-16 10:00:00
Manuel	Vega	Colesterol alto	Rango Normal	2024-08-18 10:00:00

Tabla 16: Pacientes con resultados más recientes anormales y que han sido sometidos a más de una prueba.

5.9. Encontrar los médicos que han solicitado pruebas para pacientes que tienen más de una prueba con resultados anormales

Consulta Principal

```
WITH Pacientes_Anormales AS (
    SELECT Paciente.ID_paciente, COUNT(Resultado.ID_resultado) AS Numero_Resultados_Anormales
    FROM Paciente
    JOIN Prueba ON Paciente.ID_paciente = Prueba.ID_paciente
    JOIN Resultado ON Prueba.ID_prueba = Resultado.ID_prueba
    WHERE Resultado.Valor_resultado <> Resultado.Valor_normal
    GROUP BY Paciente.ID_paciente
    HAVING COUNT(Resultado.ID_resultado) > 1
)
SELECT
    Medico.Nombre AS nombre_medico,
    Medico.Apellido AS apellido_medico,
    COUNT(DISTINCT Pacientes_Anormales.ID_paciente) AS Numero_Pacientes
FROM Medico
JOIN Prueba ON Medico.ID_medico = Prueba.ID_medico
JOIN Pacientes_Anormales ON Prueba.ID_paciente = Pacientes_Anormales.ID_paciente
GROUP BY Medico.Nombre, Medico.Apellido
ORDER BY apellido_medico;
```

Explicación de la consulta SQL

La consulta permite identificar a los médicos que tienen pacientes con resultados anormales en sus pruebas médicas. Podemos diferenciar una consulta principal y una subconsulta:

Subconsulta "pacientes_anormales": Esta parte selecciona los pacientes que tienen más de un resultado anormal en sus pruebas médicas. Los pacientes se agrupan por su ID mediante el uso de `GROUP BY` y se cuenta el número de resultados anormales mediante la funcióno de agregación `COUNT()`. Solo se incluyen aquellos pacientes que tienen más de un resultado anormal usando `HAVING`.

Consulta principal: Esta parte permite seleccionar (mediante el uso de `SELECT`,) los nombres y apellidos de los médicos, y cuenta el número de pacientes con más de un resultado anormal para cada médico (`COUNT()`). Posteriormente, se lleva a cabo la combinación de las **Medico**, **Prueba** y **Pacientes_Anormales**. Los médicos se agrupan por su nombre y apellido mediante el uso de `GROUP BY`, y los resultados se ordenan por apellido mediante el uso de `ORDER BY`.

Resultado de la consulta

Como podemos observar en la Tabla 17, la consulta muestra los nombres y apellidos de los médicos, junto con el número de pacientes con resultados anormales en sus pruebas médicas.

Interpretación

En este caso podemos observar que el médico **Dr. Keith Sunny** tiene dos pacientes con resultados anormales, mientras que el resto de médicos solo presentan un paciente con resultados anormales cada uno. Esto puede ser útil para conocer los médicos con pacientes más críticos o con resultados anormales.

nombre_medico	apellido_medico	numero_pacientes
Dra. Lucy	Delight	1
Dr. John	Galea	1
Dr. Tio	Guay	1
Dra. Laura	Hernandez	1
Dr. Carlos	Lopez	1
Dra. Juana	Lopez	1
Dra. Ana	Martinez	1
Dra. Pepi	Santana	1
Dr. Keith	Sunny	2

Tabla 17: Médicos que han solicitado pruebas para pacientes con más de un resultado anormal.

6. Triggers y procedimientos almacenados

Los triggers y procedimientos almacenados son fundamentales para automatizar tareas y mantener la integridad de los datos en la base de datos. Algunos ejemplos de apliación en nuestra base de datos de pruebas hospitalarias son los siguientes.

El código SQL de cada una de las triggers que se describen a continuación se encuentra en el archivo `TRIGGERS.sql`.

6.1. Trigger para notificar cada vez que se inserta un nuevo paciente en la tabla Paciente.

A continuación se muestra el código y los pasos a seguir para realizar un trigger en PostgreSQL que permita notificar automáticamente cada vez que se inserta un nuevo paciente en la tabla `Paciente`.

Función para el trigger

```
DROP FUNCTION IF EXISTS informar_paciente_insertado() CASCADE;

CREATE OR REPLACE FUNCTION informar_paciente_insertado()
RETURNS TRIGGER AS $$
BEGIN
    RAISE NOTICE 'Nuevo paciente insertado: Nombre=%, Apellido=%, DNI=%',
    NEW.Nombre, NEW.Apellido, NEW.DNI;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Lo que primero se hace es eliminar cualquier función que pueda existir con el mismo nombre que vamos a asignar a la función (en nuestro caso se denomina `informar_paciente_insertado`). Posteriormente, se crea la función, que está escrita en lenguaje PL/pgSQL. Esta función genera una notificación (`RAISE NOTICE`) con información sobre el nuevo paciente insertado, incluyendo el nombre, apellido y DNI. La función está asociada a un trigger que se ejecutará cada vez que se inserta una nueva fila en la tabla `Paciente`.

Definición del trigger

```
DROP TRIGGER IF EXISTS trigger_informar_paciente_insertado ON Paciente;

CREATE TRIGGER trigger_informar_paciente_insertado
AFTER INSERT ON Paciente
FOR EACH ROW
EXECUTE FUNCTION informar_paciente_insertado();
```

En este bloque de código se elimina cualquier trigger existente denominado `trigger_informar_paciente_insertado` en la tabla `Paciente`. Posteriormente, se crea un nuevo trigger con ese mismo nombre. El trigger se ejecutará después de que se realice una inserción en la tabla `Paciente`, y se invocará la función `informar_paciente_insertado`, descrita anteriormente.

Verificación del funcionamiento del trigger

Para verificar el funcionamiento del trigger, procedemos a realizar una insercción más en la tabla `Paciente` de nuestra base de datos de Pruebas Hospitalarias.

```
INSERT INTO Paciente (Nombre, Apellido, DNI, Direccion, Telefono, Email, Fecha_nacimiento, Sexo
    , no_seguridad_social)
VALUES ('Gemma', 'Gomez', '17895278A', 'Calle Rosales', '555125656', NULL, '1985-01-21', 'F', '
    123444449');
```

Tras la inserción de los datos, se indica una notificación (`NOTICE`) generada por el trigger que muestra los detalles del nuevo paciente insertado.

Salida

```
NOTICE: Nuevo paciente insertado: Nombre=Gemma, Apellido=Gomez, DNI=17895278A
```

Interpretación

Este trigger permite informar sobre nuevos pacientes insertados de una manera rápida y automática, siendo de gran utilidad en bases de datos de Pruebas Hospitalarias ya que permite asegurar un monitoreo en tiempo real y optimizando los procesos.

6.2. Trigger para evitar la modificación de la fecha_emision a una fecha futura en la tabla Resultados

A continuación se muestra el código para crear una función de trigger y un trigger en la tabla `Resultados`, que impide modificar la columna `fecha_emision` a una fecha futura.

Función para el trigger:

```
DROP FUNCTION IF EXISTS verificar_fecha_emision CASCADE;

CREATE OR REPLACE FUNCTION verificar_fecha_emision()
RETURNS TRIGGER AS $$
BEGIN
       IF NEW.fecha_emision > CURRENT_DATE THEN
       RAISE EXCEPTION 'La fecha_emision no puede ser una fecha futura: %',
       NEW.fecha_emision;
    END IF;

    RAISE NOTICE 'VALOR OLD: %', OLD.fecha_emision;
    RAISE NOTICE 'VALOR NEW: %', NEW.fecha_emision;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Esta función `verificar_fecha_emision` comprueba si la nueva fecha de emisión es mayor que la fecha actual. Si es así, lanza una excepción con un mensaje descriptivo.

Definición del trigger:

```
DROP TRIGGER IF EXISTS t_verificar_fecha_emision ON Resultado;

CREATE TRIGGER t_verificar_fecha_emision
BEFORE UPDATE OF fecha_emision ON Resultado
FOR EACH ROW
EXECUTE FUNCTION verificar_fecha_emision();
```

Este bloque de código permite eliminar cualquier trigger existente con el nombre `t_verificar_fecha_emision` en la tabla `Resultados`. Posteriormente, se crea un nuevo trigger con ese mismo nombre. El trigger se ejecutará `ANTES` de cada actualización de la columna `fecha_emision` en la tabla `Resultados`. El trigger invocará la función `verificar_fecha_emision` para realizar la verificación necesaria.

Verificación del funcionamiento del trigger

1. Ejemplo de modificación válida:

   ```
   UPDATE Resultado SET fecha_emision = '2023-11-24' WHERE id_resultado = 1;
   SELECT * FROM Resultado WHERE id_resultado = 1;
   ```

   En este caso, la fecha de emisión a la que se actualiza es una fecha pasada (o presente), no hay error. Obtenemos el siguiente mensaje:

   ```
   NOTICE: VALOR OLD: 2024-07-01 10:00:00
   NOTICE: VALOR NEW: 2023-11-24 00:00:00
   ```

2. Ejemplo de modificación no válida:

   ```
   UPDATE Resultado SET fecha_emision = '2025-01-01' WHERE id_resultado = 1;
   ```

   En este caso, como la fecha de emisión es una fecha futura, se espera que el sistema lance una excepción con el siguiente mensaje:

   ```
   ERROR: La fecha_emision no puede ser una fecha futura: 2025-01-01 00:00:00
   ```

Interpretación

Este trigger asegura que la columna `fecha_emision` en la tabla `Resultados` no pueda ser modificada a una fecha futura, garantizando la integridad temporal de los datos en nuestra base de datos de Pruebas Hospitalarias.

6.3. Trigger para registrar cambios en los datos de contacto de los pacientes

A continuación se muestra el código para crear una tabla de log, una función de trigger y un trigger en la tabla `Paciente`, que registra los cambios en los datos de contacto de los pacientes. Cabe destacar que esta tabla de log (a la que se ha denominado tabla `Log_Cambios` se ha creado exclusivamente para mostrar el funcionamiento del trigger. Esta tabla no forma parte de la estructura relacional de nuestra base de datos de Pruebas Hospitalarias y no tiene relación con otras tablas. Su propósito es demostrar cómo se pueden registrar los cambios en los datos de contacto de los pacientes mediante el uso de triggers.

Paso 1: Crear la tabla Log_Cambios

```
CREATE TABLE Log_Cambios (
ID_log SERIAL PRIMARY KEY,
ID_paciente INT NOT NULL,
campo_modificado TEXT NOT NULL,
valor_anterior TEXT NOT NULL,
valor_nuevo TEXT NOT NULL,
fecha TIMESTAMP DEFAULT NOW()
);
```

Esta tabla `Log_Cambios` almacenará los registros de los cambios en los datos de contacto de los pacientes.

Paso 2: Crear la función del trigger

```
CREATE OR REPLACE FUNCTION registrar_cambios_contacto()
RETURNS TRIGGER AS $$
BEGIN
    IF OLD.telefono IS DISTINCT FROM NEW.telefono OR OLD.email IS DISTINCT FROM NEW.email THEN
        INSERT INTO Log_Cambios (ID_paciente, campo_modificado, valor_anterior, valor_nuevo,
            fecha)
        VALUES (
        NEW.ID_paciente, 'Contacto',
            CONCAT('Tel: ', OLD.telefono, ', Email: ', OLD.email),
            CONCAT('Tel: ', NEW.telefono, ', Email: ', NEW.email),
            NOW() );
    END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Esta función `registrar_cambios_contacto` verifica si hubo cambios en el teléfono o el email del paciente. Si es así, inserta un registro en la tabla `Log_Cambios` con los detalles del cambio. En caso de no haber cambios en esos campos, la actualización de la tabla `Paciente` se llevará a cabo, pero no quedará registrada en la tabla `Log_Cambios`.

Paso 3: Crear el trigger

```
CREATE TRIGGER registrar_cambios_contacto
AFTER UPDATE ON Paciente
FOR EACH ROW
EXECUTE FUNCTION registrar_cambios_contacto();
```

Este código crea un trigger que se ejecutará `DESPUÉS` de cada actualización en la tabla `Paciente`. El trigger invocará la función `registrar_cambios_contacto` para registrar los cambios en los datos de contacto.

Paso 4: Verificación del funcionamiento del trigger

1. Ejemplo de modificación válida:

```
UPDATE Paciente SET telefono = '126896347', email = 'new@hotmail.com' WHERE
    ID_paciente = 1;

SELECT * FROM Paciente WHERE ID_paciente = 1;
SELECT * FROM Log_Cambios WHERE ID_paciente = 1;
```

   En este caso, se actualizan el teléfono y el email de un paciente, lo cual queda registrado en la tabla `Log_Cambios`. Esto lo podemos observar haciendo `SELECT` de la tabla `Log_Cambios`.

2. Ejemplo de modificación no válida:

```
UPDATE Paciente SET nombre = 'Aurora Ruiz' WHERE ID_paciente = 2;

SELECT * FROM Paciente WHERE ID_paciente = 2;
SELECT * FROM Log_Cambios WHERE ID_paciente = 2;
```

   En este caso, como solo se ha actualizado el nombre del paciente y no el teléfono o el email, no se añadirá un nuevo registro en la tabla `Log_Cambios` como podemos observar con `SELECT` (aunque si que se modifica la tabla `Paciente`).

Interpretación

Este trigger permite registrar y notificar sobre los cambios en los datos de contacto de los pacientes, concretamente en el teléfono o email, asegurando un seguimiento detallado de las modificaciones en la base de datos.

Referencias

[1] Canva. (s.f.). *Herramienta en línea para crear diagramas*. Recuperado de `https://www.canva.com/design/`

[2] Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7ª ed.). Pearson.

[3] PostgreSQL Global Development Group. (s.f.). *SQL CREATE TRIGGER*. Recuperado de `https://www.postgresql.org/docs/current/sql-createtrigger.html`

[4] LearnSQL. (s.f.). *25 Advanced SQL Query Examples*. Recuperado de `https://learnsql.com/blog/25-advanced-sql-query-examples/`

[5] dbdiagram.io. (s.f.). *Herramienta en línea para crear diagramas de base de datos*. Recuperado de `https://dbdiagram.io/home`