INFORME PROYECTO FINAL

BASE DE DATOS SABERLEER

Trabajo presentado por:

Brayan Alexander Moreno Cupitra

Trabajo presentado a:

Claudia Liliana Hernández García

UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS

FACULTAD TECNOLOGICA

BASES DE DATOS DISTRIBUIDAS

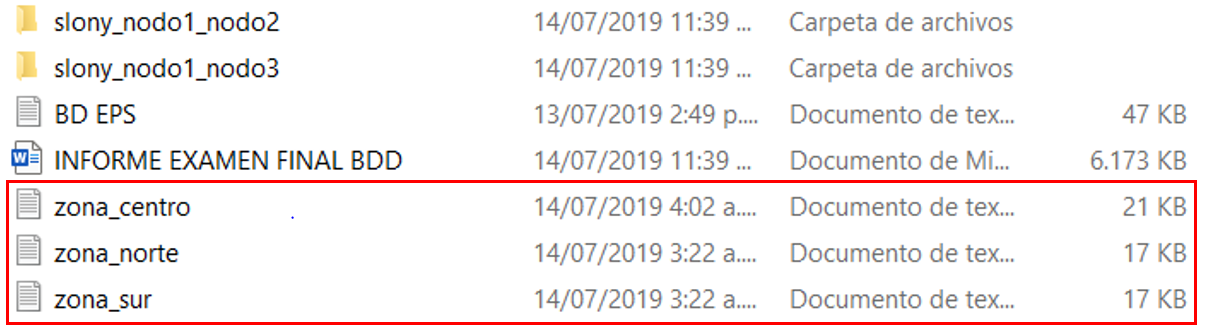
BOGOTAA D.C

2020

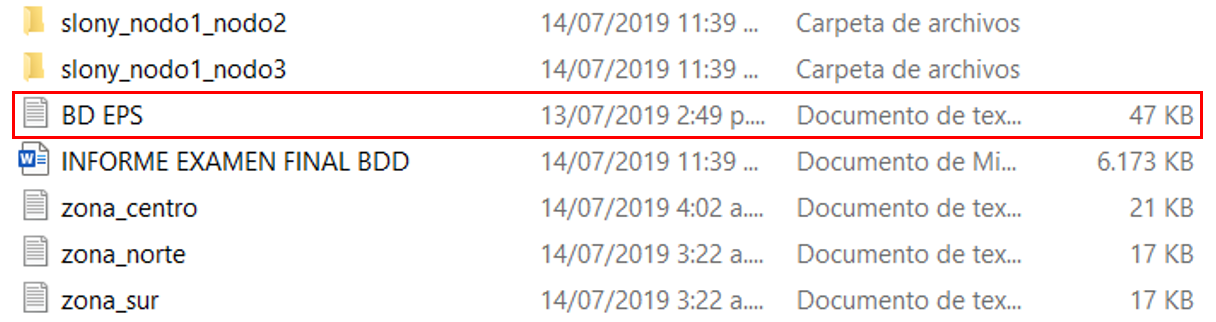
**INFORME PROYECTO FINAL**

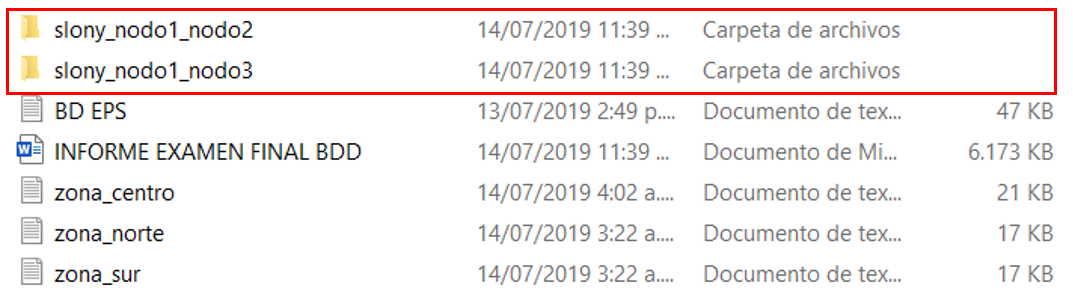
**BASE DE DATOS SABERLEER**

**\*\*----------Base de Datos y archivos para el proyecto----------\*\***

Para el desarrollo del proyecto, se utilizarán dos bases de datos “libreria\_centro”, “libreria\_sur” y “libreria\_norte”. Los archivos .txt que contienen los scripts de las bases de datos mencionadas anteriormente tienen por nombre “zona\_centro.txt”, “zona\_norte.txt” y “zona\_sur.txt” respectivamente, y se encuentran en la carpeta “EXAMEN FINAL BDD” del archivo comprimido del presente informe:

Los scripts contienen el DDL de las bases de datos y la inserción de los datos.

La base de datos completa de la EPS (sin fragmentación ni replicación) se encuentra en el archivo “BD EPS.txt” y tiene por nombre “eps”:

Para el proceso de replica se utilizarán los archivos de las carpetas “slony\_nodo1\_nodo2” y “slony\_nodo1\_nodo3”:

**\*\*----------Información de las maquinas con las cuales se va a trabajar----------\*\***

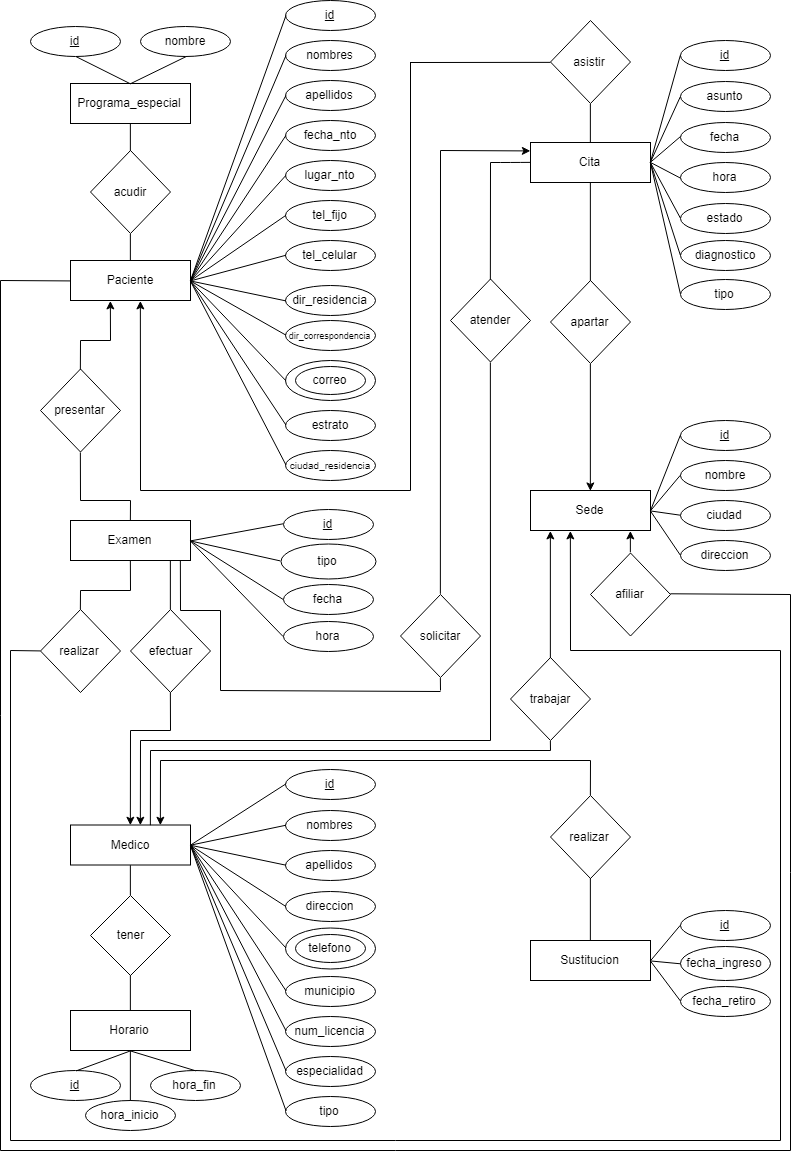
Para el desarrollo del proyecto se van a utilizar 3 computadores(nodos) diferentes. La explicación correspondiente a la distribución de los datos se encuentra mas adelante en el presente informe.

Las direcciones IP de los ordenadores son las siguientes:

* **Nodo 1:** 192.168.0.3
* **Nodo 2:** 192.168.0.5
* **Nodo 3:** 192.168.0.11

Las bases de datos van a estar distribuidas en los nodos de la siguiente manera:

* **Nodo 1:** Contendrá la base de datos llamada “eps\_centro”
* **Nodo 2:** Contendrá la base de datos llamada “eps\_norte”
* **Nodo 3:** Contendrá la base de datos llamada “eps\_sur”

1. **Diseño de la base de datos centralizada**
2. **Diagrama Entidad-Relación**

**Aclaraciones sobre el diagrama**

A continuación, se hace una explicación de las entidades utilizadas, sus atributos, las relaciones entre entidades y la justificación de los atributos añadidos por nosotros, es decir, aquellos que no se mencionan en el enunciado pero que determinamos que son apropiados.

1. **Entidades y atributos**

* **Paciente:** Esta entidad identifica a todos los pacientes que están afiliados a la EPS. Consta de los siguientes atributos: id, nombres, apellidos, fecha\_nto (fecha de nacimiento), tel\_fijo (telefono fijo), tel\_celular (telefono celular), dir\_residencia (direccion de residencia), dir\_correspondencia (direccion de residencia), correo (atributo multivaluado) y ciudad\_residencia (ciudad de residencia).
* **Programa\_especial:** Esta entidad identifica a aquellos programas especiales ofrecidos por la EPS y que son tomados por aquellos pacientes que son adultos mayores.

**Atributos adicionales:**

* **id:** Numero que identifica únicamente a cada programa.
* **nombre:** Nombre del programa
* **Medico:** Esta entidad representa a todos los médicos que pertenecen y trabajan en la EPS. Consta de los siguientes atributos: id, nombres, apellidos, direccion, telefono (atributo multivaluado), municipio, num\_licencia (número de licencia médica), especialidad y tipo (representa el tipo de medico: titular, interino o sustituto).
* **Horario:** Esta entidad representa los horarios en los cuales los médicos trabajan. **Atributos adicionales:**
* **id:** Numero que identifica únicamente a cada horario por día, es decir, a cada jornada laboral,
* **hora\_inicio:** Hora de inicio de la jornada laboral.
* **hora\_fin:** Hora de fin de la jornada laboral.
* **Sustitución**: Esta entidad representa todas aquellas sustituciones que pueda tener un médico (una o varias).

**Atributos adicionales:**

* **id:** Numero que identifica únicamente a cada sustitución, con lo cual, un médico puede tener más de una.
* **fecha\_ingreso:** Fecha de ingreso del médico.
* **fecha\_retiro:** Fecha de retiro del médico.
* **Sede:** Esta entidad representa las sedes con las que cuenta la EPS en las diferentes ciudades del país.

**Atributos adicionales:**

* **id:** Numero que identifica únicamente a cada sede.
* **nombre:** Nombre que identifica a cada sede.
* **ciudad**: Ciudad en la que está ubicada la sede.
* **direccion:** Dirección en la que está ubicada la sede.
* **Cita:** Esta entidad representa todas aquellas citas que son reservadas en la EPS.

**Atributos adicionales:**

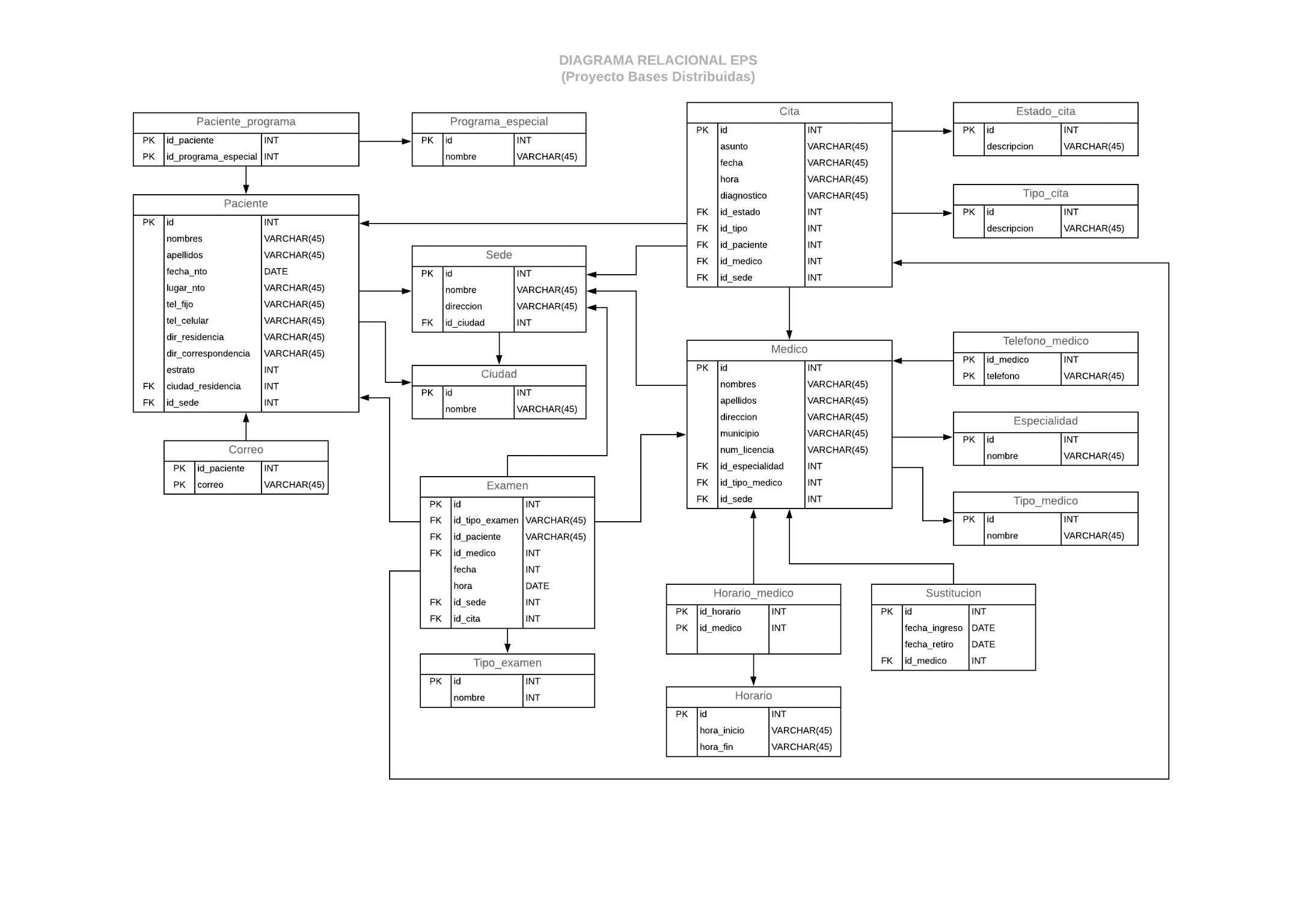
* **id:** Numero que identifica únicamente a cada cita.
* **asunto:** Representa el motivo por el cual se reservó la cita.
* **fecha**: Fecha de la cita.
* **hora:** Hora de la cita.
* **estado:** Representa el estado de la cita (Atendido, No atendido, Cancelado).
* **diagnostico:** Representa el diagnóstico que genera el medico sobre el estado de salud del paciente.
* **tipo:** Representa el tipo de cita: medicina general, especialidad o urgencias.
* **Examen:** Esta entidad representa los exámenes que le son solicitados a los pacientes por parte de los médicos,

**Atributos adicionales:**

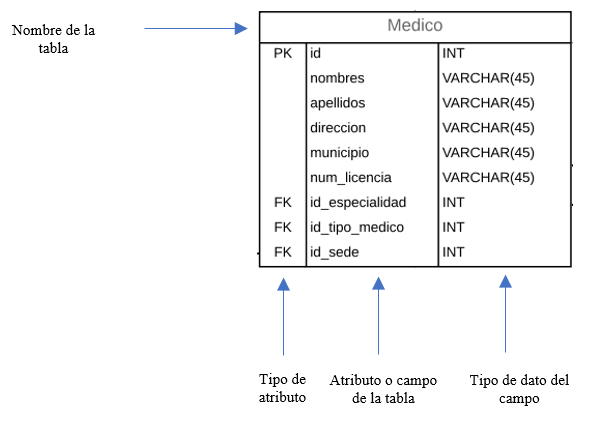
* **id:** Numero que identifica únicamente a cada examen.
* **tipo:** Representa el tipo de examen que se realiza el paciente (por ejemplo: electrocardiograma, radiografía, ecografía, TAC etc.).
* **fecha**: Fecha del examen.
* **hora:** Hora del examen.

1. **Relaciones**

* **acudir:** Relación entre las entidades **Paciente** y **Programa\_especial,** que representa aquellos pacientes que acuden o están yendo a uno o más programas especiales que ofrece la EPS.
* **Cardinalidad:** M:N (Muchos a muchos). Un paciente puede acudir a uno o más programas especiales y, a un programa especial pueden acudir uno o más pacientes.
* **tener:** Relación entre las entidades **Medico** y **Horario,** que representa los horarios o jornadas de trabajo que tiene cada médico, horario que puede ser diferente cada día de la semana.
* **Cardinalidad:** M:N (Muchos a muchos). Un médico puede tener uno o más horarios y un horario lo puede tener varios médicos.
* **asistir:** Relación entre las entidades **Paciente** y **Cita,** que representa las citas a las que asiste un paciente, es decir, las citas que tiene asignadas.
* **Cardinalidad:** 1:N (Uno a muchos). Un paciente puede asistir a varias citas, pero, a una cita puede asistir un único paciente, es decir, las citas son individuales.
* **atender:** Relación entre las entidades **Medico** y **Cita,** que representa los médicos que atienden las citas, es decir, qué medico atiende qué cita.
* **Cardinalidad:** 1:N (Uno a muchos). Un médico puede atender varias citas, pero, una cita puede ser atendida por un único médico.
* **apartar:** Relación entre las entidades **Sede** y **Cita,** que representa las citas que son apartadas en las distintas sedes.
* **Cardinalidad:** 1:N (Uno a muchos). En una sede pueden estar apartadas una o más citas, pero, una cita, puede estar apartada en una única sede.
* **trabajar:** Relación entre las entidades **Medico** y **Sede,** que representa los médicos que trabajan u ofrecen atención médica en cada una de las sedes.
* **Cardinalidad:** N:1 (Muchos a uno). Un médico trabaja en una única sede y en una sede pueden trabajar muchos médicos.
* **afiliar:** Relación entre las entidades **Paciente** y **Sede,** que representa los pacientes que están afiliados en cada una de las sedes de la EPS que se encuentran en las diferentes ciudades.
* **Cardinalidad:** N:1 (Muchos a uno). Un paciente está afiliado en una única sede, pero, en una sede pueden estar afiliados muchos pacientes.
* **realizar (entre Medico y Sustitucion):** Relación entre las entidades **Medico** y **Sustitucion,** que representa las sustituciones que puede tener o realizar un médico.
* **Cardinalidad:** 1:N (Uno a muchos). Un médico puede realizar o tener una o más sustituciones y una sustitución la puede realizar un solo médico.
* **realizar (entre Examen y Sede):** Relación entre las entidades **Examen** y **Sede,** representalos exámenes médicos que son realizados en cada sede, es decir, en donde se realiza cada examen.
* **Cardinalidad:** N:1 (Muchos a uno). Un examen se puede realizar en una sede en específico y en una sede se pueden realizar muchos exámenes.
* **efectuar:** Relación entre las entidades **Examen** y **Medico,** representalos exámenes realizados por los médicos, es decir, qué médico realiza qué examen.
* **Cardinalidad:** N:1 (Muchos a uno). Un examen puede ser realizado por un único médico y un médico puede realizar exámenes.
* **presentar:** Relación entre las entidades **Paciente** y **Examen,** representalos exámenes que se realizan los pacientes.
* **Cardinalidad:** 1:N (Uno a muchos). Un paciente puede realizarse muchos exámenes y un examen se lo realiza un solo paciente.
* **solicitar:** Relación entre las entidades **Cita** y **Examen,** representalos exámenes que son solicitados a los pacientes en cada cita.
* **Cardinalidad:** N:M (Muchos a muchos). En una cita se le puede solicitar a un paciente que se realice muchos exámenes y un examen puede ser solicitado en muchas citas.

1. **Diagrama Relacional**

**Aclaraciones sobre el diagrama**

1. **Notación:** A continuación, se explica mediante un ejemplo, la notación utilizada para las tablas en el diagrama relacional. Por ejemplo, tomemos la tabla *Medico* del diagrama anterior:

La tabla está dividida en una parte superior en la cual se indica el nombre de tabla, y una parte inferior que está dividida a su vez en tres partes: tipo de atributo, atributo o campo de la tabla y tipo de dato del campo.

* **Tipo de atributo:** Representa el tipo de campo, es decir: llave primeria (PK), llave foránea (FK) o campo normal (en este caso no lleva las siglas PK un FK).
* **Atributo o campo:** Indica el nombre del campo.
* **Tipo de dato de campo:** Indica el tipo de dato del campo (INT, VARCHAR, DATE, etc.).

1. **Tablas:** A continuación, se hacen algunas aclaraciones acerca del diagrama relacional:

* La tabla **Horario\_medico** es el resultado de la relación muchos a muchos (N:M) **tener,** presente en el diagrama Entidad-Relación, por lo cual, la tabla no tiene como nombre **tener,** sino que se renombra como **Horario\_medico.**
* La tabla **Paciente\_programa** es el resultado de la relación muchos a muchos (N:M) **acudir,** presente en el diagrama Entidad-Relación, por lo cual, la tabla no como nombre **acudir,** sino que se renombra como **Paciente\_programa.**

1. **Diseño del ambiente distribuido.**

Para la implementación de la base de datos y los diagramas tenemos como albergue las sedes de Bogotá, Medellín, Cali, Barranquilla, Cartagena, Cúcuta, Villavicencio, Pasto y Leticia, las cuales se encuentran repartidas en tres nodos de la siguiente manera:



**Zona Central del país (Nodo1: base de datos “eps\_centro”):**

* Medellín
* Bogotá
* Villavicencio

**Zona Norte del país (Nodo 2: base de datos “eps\_norte”):**

* Barranquilla
* Cartagena
* Cúcuta

**Zona Sur del país (Nodo 3: base de datos “eps\_sur”):**

* Leticia
* Pasto
* Cali

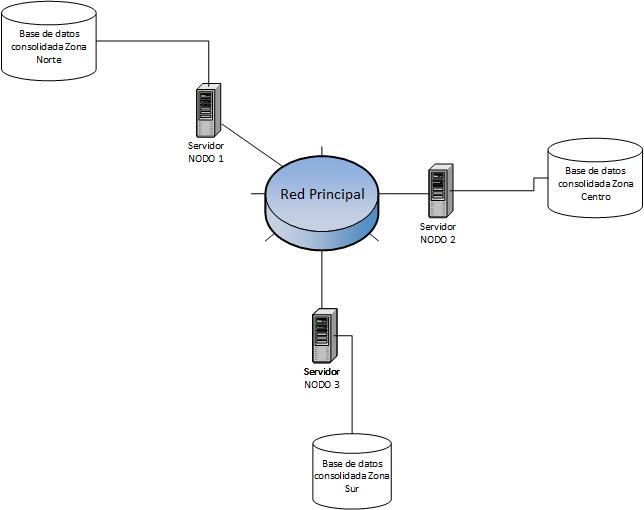


Figura 1. Esquema del ambiente distribuido

**Sistemas Operativos y Motores de Bases de Datos**

Los nodos se encontrarán soportados por el sistema operativo Windows 10 y Windows 8, debido a que son los sistemas operativos que se encuentran instalados en las salas de informática de la facultad.

El DBMS a utilizar para la base de datos será PostgresSQL en su versión 9.5, ya que es la que comparten la mayoría de los equipos de las aulas de informática del bloque 13.

1. **Diseño de la fragmentación (vertical, horizontal y mixta) y de la réplica**

A continuación, se da una explicación del diseño de la fragmentación y de la réplica de la base de datos, la cantidad de nodos en los cuales va a estar distribuida la base de datos, el esquema de la base de datos en cada uno de los nodos y la distribución de los datos en cada uno de ellos.

* **Cantidad de nodos:** Los datos de la base de datos se fragmentarán en tres nodos, es decir, que estarán repartidos en tres máquinas diferentes.
* **Tipo de fragmentación:** Para la base de datos se utilizará únicamente fragmentación horizontal.

**Tablas a replicar**

* Autor
* Editorial
* Libro

**Tablas a fragmentar**

* Sucursal
* Sucursal\_Libro
* Venta
* Ciudad
* Empleado (por cargo)

**DISEÑO DE LA RÉPLICA**

**Conceptos a tener en cuenta para la replica:**

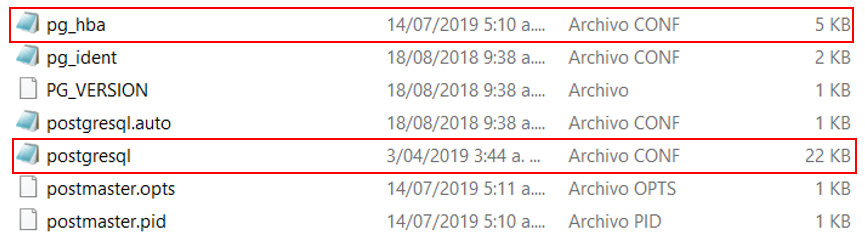
* **Nodo maestro:** Representa el ordenador o maquina en donde van a estar los datos que se van a replicar en los demás nodos, denominados nodos esclavo.
* **Nodo esclavo:** Representa los ordenadores o maquinas en los cuales se van a replicar los datos.

En nuestro caso el nodo maestro va a ser el **Nodo 1 (base de datos “librería\_centro”)** y los datos de este se van a replicar en los nodos **Nodo 2 (base de datos “libreria\_Sur”)** y **Nodo3 (base de datos “libreria\_norte”)**, que serían los nodos esclavos.

**Configuración de archivos de PostgreSQL**

En este paso debemos buscar ruta de instalación de PostgreSQL, la versión y ubicamos la carpeta **data:**

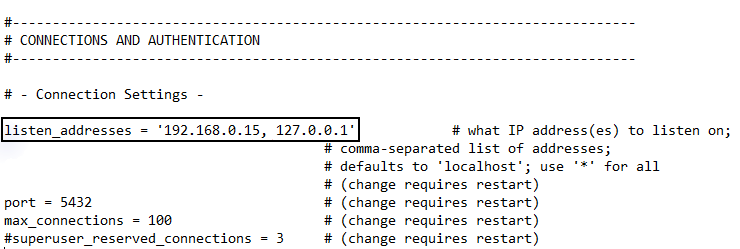
*C:\Program Files\PostgreSQL\10\data*

Luego, buscamos los siguientes archivos: *postgresql.conf y pg\_hba.conf*:

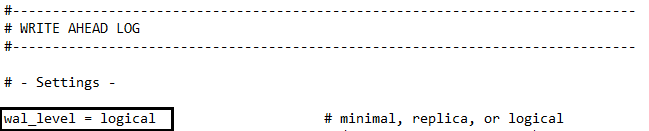
En el archivo *postgresql.conf* buscamos la opción “listen\_addresses” que se encuentra en el apartado CONNECTIONS AND AUTHENTICATION, y agregamos las ip’s de los nodos remotos, esto hace con el fin de darles acceso a nuestra maquina:

*listen\_addresses = ‘ip1, ip2, ip3’*

De esta manera, se indica que el motor de bases de datos PostgreSQL va a “escuchar” o permitir que estas direcciones IP (ordenador) externas se conecten a él. La siguiente imagen muestra lo anteriormente explicado:



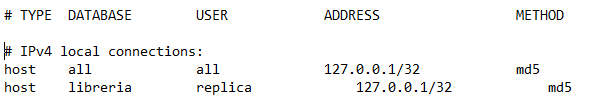
Luego, buscamos la opción “wal\_level” que se encuentra en el apartado WRITE AHEAD LOG, se le quita el comentario (#) y le asigna el valor de logical:

**

De esta manera, se indica que el motor de bases de datos PostgreSQL va a permitir la replicacion de tablas.

**NOTA:** La configuración del archivo *postgresql.conf* se debe realizar para cada uno de los nodos, tanto para el nodo maestro como para los dos nodos esclavos. Adicionalmente, para que los cambios surtan efecto, se debe reiniciar PostgreSQL.

En el archivo *pg\_hba.conf* debemos asegurarnos de colocar las direcciones IP tanto de la maquina maestro como de las maquinas esclavo. Para ello debemos ubicarnos en el apartado **IPv4 local connections:** y agregar en una línea cada una de las direcciones IP correspondientes. También agregamos otra línea con la dirección local, pero especificamos la base de datos y el usuario, esto se hace para permitir que el usuario “replica” pueda crear replicaciones sobre las tablas de la base de datos “libreria”.



**NOTA:** La configuración del archivo *pg\_hba.conf* se debe realizar para cada uno de los nodos, tanto para el nodo maestro como para los dos nodos esclavos. Adicionalmente, para que los cambios surtan efecto, se debe reiniciar PostgreSQL.

Una vez hecho todo lo anteriormente explicado, ya estamos listos para empezar con el proceso de réplica de datos

**PROCESO DE RÉPLICA DE DATOS**

**IMPORTANTE:** Los datos se van a replicar primero desde el **nodo 1** (base de datos “libreria\_centro”) al **nodo 2** (base de datos “libreria\_sur”) y luego desde el **nodo 1** (base de datos “libreria\_centro”) al **nodo 3** (base de datos “libreria\_sur”).

Las tablas que se van a replicar son:

* Autor
* Editorial
* Libro
* Cliente

Debido a que se realiza una réplica de datos, el esquema de la base de datos para cada nodo es el mismo, para cada una de las tablas de la lista anterior:

|  |  |  |
| --- | --- | --- |
| **Nodo1** | **Nodo2** | **Nodo3** |
|  |  |  |

Lo primero que debemos hacer es crear las bases de datos en cada uno de los nodos, de la siguiente manera:

* **Nodo 1:**

create database libria\_centro;

\c libreria\_centro

* **Nodo 2:**

create database libreria\_sur;

\c libreria\_sur

* **Nodo 3:**

create database librería\_norte;

\c librería\_norte

EL DDL para cada uno de los nodos y para las tablas que se van a replicar es el mismo y queda de la siguiente manera:

create table telefono\_empleado(

telefono varchar(20) unique not null,

codigo\_empleado int not null,

foreign key(codigo\_empleado) references empleado(codigo),

primary key(telefono,codigo\_empleado)

);

create table sexo(

codigo int not null,

sexo varchar(2) not null,

primary key(codigo)

);

create table autor(

identificacion int not null,

nombre varchar(50) not null,

e\_mail varchar(50),

codigo\_sexo int,

foreign key(codigo\_sexo) references sexo(codigo),

primary key(identificacion)

);

create table telefono\_autor(

telefono varchar(20) unique not null,

identificacion\_autor int not null,

foreign key(identificacion\_autor) references autor(identificacion),

primary key(telefono,identificacion\_autor)

);

create table ciudad(

codigo int not null,

ciudad varchar(25) not null,

primary key(codigo)

);

create table sucursal(

nombre varchar(50) not null,

ciudad int not null,

primary key(nombre),

foreign key(ciudad) references ciudad(codigo)

);

create table telefono\_sucursal(

telefono varchar(20) unique not null,

nombre\_sucursal varchar(50) not null,

foreign key(nombre\_sucursal) references sucursal(nombre),

primary key(telefono, nombre\_sucursal)

);

create table cargo(

codigo int not null,

cargo varchar(40) not null,

primary key(codigo)

);

create table empleado(

codigo int not null,

documento int not null unique,

nombre varchar(30) not null,

salario real not null,

fecha\_de\_nacimiento date not null,

fecha\_de\_contratacion date not null,

codigo\_cargo int not null,

nombre\_sucursal varchar(40) not null,

foreign key(nombre\_sucursal) references sucursal(nombre),

primary key(codigo)

);

create table venta(

codigo int not null,

codigo\_sucursal\_libro int not null,

identificacion\_cliente int not null,

fecha\_compra date not null DEFAULT CURRENT\_DATE,

primary key(codigo),

foreign key(codigo\_sucursal\_libro) references sucursal\_libro(codigo),

foreign key(identificacion\_cliente) references cliente(identificacion)

);

create table telefono\_cliente(

telefono varchar(20) not null,

identificacion\_cliente int not null,

foreign key(identificacion\_cliente) references cliente(identificacion),

primary key(telefono,identificacion\_cliente)

);

create table telefono\_autor(

telefono varchar(20) unique not null,

identificacion\_autor int not null,

foreign key(identificacion\_autor) references autor(identificacion),

primary key(telefono,identificacion\_autor)

);

create table editorial(

codigo int not null,

nombre varchar(60) not null,

primary key(codigo)

);

create table libro(

codigo int not null,

isbn varchar(13) not null unique,

titulo text not null,

nro\_paginas int not null,

editorial int not null,

identificacion\_autor int not null,

foreign key(editorial) references editorial(codigo),

foreign key(identificacion\_autor) references autor(identificacion),

primary key(codigo)

);

create table sucursal\_libro(

codigo int not null,

nombre\_sucursal varchar(50) not null,

codigo\_libro int not null,

cantidad\_ejemplares int not null,

primary key(codigo),

foreign key(nombre\_sucursal) references sucursal(nombre),

foreign key(codigo\_libro) references libro(codigo)

);

Una vez creadas las bases de datos y creadas las tablas con el DDL anterior, insertamos los datos correspondientes a las tablas anteriormente listadas solo el **nodo 1** (base de datos “libreria\_centro”), ya que desde este nodo se van a replicar los datos a los otros dos:

insert into ciudad values (1, 'Bogota'), (2, 'Medellin');

insert into sucursal values ('Libreria Bogotá', 1), ('Libreria Medellin', 2);

insert into cargo values (1, 'Gerente'), (2, 'Vendedor');

insert into empleado values

(1000, 1024, 'Pepe', 3000000, '1990-05-12', '2020-01-15', 1, 'Libreria Bogotá'),

(2000, 1025, 'Juan', 2000000, '1992-06-11', '2020-02-15', 2, 'Libreria Bogotá'),

(3000, 1026, 'Maria', 3000000, '1991-05-14', '2020-01-18', 1, 'Libreria Medellin'),

(4000, 1027, 'Laura', 1500000, '1994-07-17', '2020-05-15', 2, 'Libreria Bogotá'),

(5000, 1028, 'Camio', 2000000, '1991-02-13', '2020-06-15', 2, 'Libreria Bogotá'),

(6000, 1029, 'Diego', 1000000, '1992-06-17', '2020-03-15', 2, 'Libreria Medellin'),

(7000, 1034, 'Isabell', 1600000, '1994-09-13', '2020-01-23', 2, 'Libreria Medellin'),

(8000, 1035, 'Carlos', 1800000, '1996-12-23', '2020-01-25', 2, 'Libreria Medellin');

insert into telefono\_empleado values (1234567,1000), (5346512,2000), (3219784,3000), (1349824,4000),

(7981423,5000), (5423917,6000), (4312890,7000), (2419383,8000);

insert into sexo VALUES (1, 'F'), (2, 'M');

insert into autor values (1, 'Mario Mendoza', 'MM@gmai.com', 1), (2, 'William Ospina', 'WO@gmai.com', 1),

(3, 'Renán Vega', 'RV@gmai.com', 1), (4, 'Soledad Acosta', 'SA@gmai.com', 2),

(5, 'Isabel Allende', 'IA@gmai.com', 2);

insert into editorial values (1, 'Planeta'), (2, 'Anagrama');

insert into sucursal\_libro VALUES (1, 'Libreria Bogotá', 1, 11), (2, 'Libreria Bogotá', 2, 8),

(3, 'Libreria Bogotá', 3, 12), (4, 'Libreria Bogotá', 4, 8), (5, 'Libreria Bogotá', 5, 8), (6, 'Libreria Bogotá', 6, 4),

(7, 'Libreria Bogotá', 7, 12), (8, 'Libreria Bogotá', 8, 11), (9, 'Libreria Medellin', 1, 11),

(10, 'Libreria Medellin', 2, 12), (11, 'Libreria Medellin', 3, 14), (12, 'Libreria Medellin', 4, 7),

(13, 'Libreria Medellin', 5, 9), (14, 'Libreria Medellin', 6, 9), (15, 'Libreria Medellin', 7, 4),

(16, 'Libreria Medellin', 8, 6);

insert into clienteP values (2044, 'Pepito', 'pepito@gmail.com', 1), (2045, 'Claudia', 'Claudia@gmail.com', 1),

(2046, 'Sofia', 'sofia@gmail.com', 2), (2047, 'Felipe', 'felipe@gmail.com', 2),

(2048, 'Karen', 'karen@gmail.com', 2), (2049, 'Juana', 'juana@gmail.com', 1);

insert into telefono\_cliente values (1244566, 1044), (7889566, 1045), (4344566, 1046), (6844566, 1047),

(9244566, 1048), (8444566, 1049);

insert into venta VALUES

(1, 1, 1044, '2019-12-12');

insert into venta VALUES (2, 2, 1044), (3, 3, 1045), (4, 9, 1048), (5, 12, 1048), (6, 11, 1046);

insert into venta VALUES (7, 13, 1047, '2019-12-24');

insert into venta values (8, 1, 1046);

insert into venta values (9, 14, 1045);

insert into venta values (10, 12, 1049);

insert into venta values (11, 14, 1044);

insert into venta values (12, 12, 1047);

insert into venta values (13, 14, 1045);

insert into venta values (14, 12, 1049);

insert into venta values (16, 1, 1045);

insert into venta values (17, 4, 1049);

insert into venta values (18, 2, 1045);

insert into venta values (19, 3, 1049);

insert into venta values (20, 16, 1046);

**Replica desde el nodo 1 al nodo 2**

Para poder realizar la réplica, necesitamos crear un nuevo rol en el **nodo 1** (base de datos “librería\_centro”):

create role replica REPLICATION LOGIN PASSWORD '123456';

Luego realizamos las publicaciones respectivas, el rol “replica” se le debe otorgar los permisos para permitir realizar las publicaciones:

GRANT all on autor to replica;

CREATE PUBLICATION public\_autor FOR TABLE autor;

GRANT all on libro to replica;

CREATE PUBLICATION public\_libro FOR TABLE libro

GRANT all on editorial to replica;

CREATE PUBLICATION public\_editorial FOR TABLE editorial;

GRANT all on cliente to replica;

CREATE PUBLICATION public\_cliente FOR TABLE cliente;

Seguidamente, en el **nodo 2** (base de datos “librería\_sur”) se crean las respectivas subscripciones a las publicaciones creada desde el nodo principal.

create subscription sub\_autor

connection 'host=127.0.0.1 port=5432 user=replica dbname=libreria password=123456'

publication public\_autor;

create subscription sub\_editorial

connection 'host=127.0.0.1 port=5432 user=replica dbname=libreria password=123456'

publication public\_editorial;

create subscription sub\_libro

connection 'host=127.0.0.1 port=5432 user=replica dbname=libreria password=123456'

publication public\_libro;

create subscription sub\_cliente

connection 'host=127.0.0.1 port=5432 user=replica dbname=libreria password=123456'

publication public\_cliente;

**Replica desde el nodo 1 al nodo 3**

Para poder realizar la réplica, necesitamos crear un nuevo rol en el **nodo 1** (base de datos “librería\_centro”):

create role replica REPLICATION LOGIN PASSWORD '123456';

Luego realizamos las publicaciones respectivas, el rol “replica” se le debe otorgar los permisos para permitir realizar las publicaciones:

GRANT all on autor to replica;

CREATE PUBLICATION public\_autor FOR TABLE autor;

GRANT all on libro to replica;

CREATE PUBLICATION public\_libro FOR TABLE libro

GRANT all on editorial to replica;

CREATE PUBLICATION public\_editorial FOR TABLE editorial;

GRANT all on cliente to replica;

CREATE PUBLICATION public\_cliente FOR TABLE cliente;

Seguidamente, en el **nodo 3** (base de datos “librería\_norte”) se crean las respectivas subscripciones a las publicaciones creada desde el nodo principal.

create subscription subs\_autor

connection 'host=127.0.0.1 port=5432 user=replica dbname=libreria password=123456'

publication public\_autor;

create subscription subs\_editorial

connection 'host=127.0.0.1 port=5432 user=replica dbname=libreria password=123456'

publication public\_editorial;

create subscription subs\_libro

connection 'host=127.0.0.1 port=5432 user=replica dbname=libreria password=123456'

publication public\_libro;

create subscription sub\_cliente

connection 'host=127.0.0.1 port=5432 user=replica dbname=libreria password=123456'

publication public\_cliente;

**Replica desde el nodo 2 al nodo 1**

Para poder realizar la réplica, necesitamos crear un nuevo rol en el **nodo 2** (base de datos “librería\_centro”):

create role replica REPLICATION LOGIN PASSWORD '123456';

Luego realizamos las publicaciones respectivas, el rol “replica” se le debe otorgar los permisos para permitir realizar las publicaciones:

GRANT all on cliente to replica;

CREATE PUBLICATION public\_cliente FOR TABLE cliente;

Seguidamente, en el **nodo 1** (base de datos “librería\_centro”) se crea la respectiva suscripción a la publicación creada desde el **nodo 2.**

create subscription subscri\_cliente

connection 'host=127.0.0.1 port=5433 user=replica dbname=libreria password=123456'

publication public\_cliente;

**Replica desde el nodo 3 al nodo 1**

Para poder realizar la réplica, necesitamos crear un nuevo rol en el **nodo 3** (base de datos “librería\_centro”):

create role replica REPLICATION LOGIN PASSWORD '123456';

Luego realizamos las publicaciones respectivas, el rol “replica” se le debe otorgar los permisos para permitir realizar las publicaciones:

GRANT all on cliente to replica;

CREATE PUBLICATION public\_cliente FOR TABLE cliente;

Seguidamente, en el **nodo 1** (base de datos “librería\_centro”) se crea la respectiva suscripción a la publicación creada desde el **nodo 3.**

create subscription subscrip\_cliente

connection 'host=127.0.0.1 port=5434 user=replica dbname=libreria password=123456'

publication public\_cliente;

**DISEÑO DE LA FRAGMENTACIÓN**

Para la fragmentación se va a tener en cuenta las sedes con las que cuenta la librería SABERLEER (Bogotá, Medellín, Cali, Pasto, Barranquilla, Cartagena, Riohacha y Santa Marta), de tal forma que los datos van a estar distribuidos en los tres nodos de la siguiente manera:

1. **Zona Norte del país (Nodo 1: base de datos “libreria\_centro”):**

* Medellín
* Bogotá

1. **Zona Sur del país (Nodo 2: base de datos “libreria\_sur”):**

* Pasto
* Cali

1. **Zona Central del país (Nodo 3: base de datos “libreria\_norte”):**

* Barranquilla
* Cartagena
* Santa Marta
* Riohacha

Se puede observar, por cada nodo van a estar alojados teniendo en cuenta su posición geográfica, es decir, agrupándolas de acuerdo a cuáles estén más cercanas unas de las otras. En el Nodo 1, por ejemplo, van a estar alojados los datos de los empleados que pertenecen a las sucursales de dicho nodo, los datos de los clientes que están afiliados a las sucursales de dicho nodo y así sucesivamente.

**Lista de tablas a fragmentar:**

**Sucursal:** La tabla *Sucursal* guarda los datos correspondientes a cada sucursl con la que cuenta la libreria. Consta de dos columnas: nombre y ciudad, como lo muestra la siguiente imagen, resultado de ejecutar la sentencia *select \* from sucursal*:

**Justificación para la fragmentación**: Puede ser de interés para la libreria guardar los datos de las sucursales con las que cuenta teniendo en cuenta su ubicación geográfica (ciudad en la que se encuentra la sucursal), es decir, agrupándolas, en este caso en 3 grupos, teniendo en cuenta cuales están más cerca una de las otras geográficamente hablando. Por ejemplo, las sucursales Barranquilla, Cartagena, Santa Marta y Riohacha se encuentran cerca entre ellas, por lo tanto, puede ser de interés para la libreria guardar los datos de las mismas en un mismo lugar (nodo).

Teniendo en cuenta lo anterior, si agrupamos las sucursales de manera geográfica, 3 nodos, por lo tanto, el primer nodo guardará los datos de las sucursales ubicadas en Bogotá y Medellín, el segundo nodo los datos de las sucursales ubicadas en Cali y Pasto, finalmente, el nodo 3 guardará los datos de las sucursales ubicadas en Barranquilla, Cartagena, Santa Marta y Riohacha.

Por tanto, la tabla *Sucursal* se puede fragmentar horizontalmente de la tal manera que las tuplas se fragmenten de acuerdo a la ciudad en la cual está ubicada cada sucursal (ciudad) y, cada uno de esos fragmentos se coloquen en cada uno de los nodos, como ya se indicó anteriormente. La fragmentación queda de la siguiente manera;

Debido a que se realiza una fragmentación horizontal, el esquema de la base de datos para cada nodo es el mismo:

|  |  |  |
| --- | --- | --- |
| **Nodo1** | **Nodo2** | **Nodo3** |
|  |  |  |

.

El DDL y el DML de la base de datos quedan de la siguiente manera:

**Para el Nodo 1: base de datos “libreria\_centro”**

create table sucursal(

nombre varchar(50) not null,

ciudad int not null,

primary key(nombre),

foreign key(ciudad) references ciudad(codigo)

);

insert into sucursal values

('Libreria Bogotá', 1),

('Libreria Medellin', 2);

**Para el Nodo 2: base de datos “libreria\_sur”**

create table sucursal(

nombre varchar(50) not null,

ciudad int not null,

primary key(nombre),

foreign key(ciudad) references ciudad(codigo)

);

insert into sucursal values

('Libreria Cali', 3),

('Libreria Pasto', 4);

**Para el Nodo 3: base de datos “libreria\_norte”**

create table sucursal(

nombre varchar(50) not null,

ciudad int not null,

primary key(nombre),

foreign key(ciudad) references ciudad(codigo)

);

insert into sucursal values

('Libreria Santa Marta', 5),

('Libreria Barranquilla', 5)

('Libreria Cartagena', 7)

('Libreria Riohacha', 8);

**Sucursal\_libro:** La tabla *sucursal\_libro* guarda los datos correspondientes de los libros disponibles en cada sucursal con sus respectivas cantidades. Consta de cuatro columnas: codigo, nombre\_sucursal, código\_libro y cantidad\_ejemplares.

**Justificación de la fragmentación:** Puede ser de interés mantener la información de los libros disponibles de forma separada teniendo en cuenta la ubicación geográfica de las sucursales a las cuales están registrados. Teniendo en cuenta esto, si un libro está disponible en la sucursal de Bogotá sus datos deberían estar en el nodo 1 y no en los otros nodos, ya que el nodo 1, como vimos anteriormente, agrupa los datos de las sucursales Bogotá y Medellín.

Por tanto, la tabla *sucrusal\_libro* se puede fragmentar horizontalmente de la tal manera que las tuplas se fragmenten de acuerdo a la sucursal en la cual está disponible el libro y, cada uno de esos fragmentos se coloque en cada uno de los nodos, como ya se indicó anteriormente.

Debido a que se realiza una fragmentación horizontal, el esquema de la base de datos para cada nodo es el mismo:

|  |  |  |
| --- | --- | --- |
| **Nodo1** | **Nodo2** | **Nodo3** |
|  |  |  |

.

El DDL de la base de datos queda de la siguiente manera:

**Para el Nodo 1: base de datos “librería\_centro”**

create table sucursal\_libro(

codigo int not null,

nombre\_sucursal varchar(50) not null,

codigo\_libro int not null,

cantidad\_ejemplares int not null,

primary key(codigo),

foreign key(nombre\_sucursal) references sucursal(nombre),

foreign key(codigo\_libro) references libro(codigo)

);

insert into sucursal\_libro VALUES (1, 'Libreria Bogotá', 1, 11), (2, 'Libreria Bogotá', 2, 8),

(3, 'Libreria Bogotá', 3, 12), (4, 'Libreria Bogotá', 4, 8), (5, 'Libreria Bogotá', 5, 8),

(6, 'Libreria Bogotá', 6, 4), (7, 'Libreria Bogotá', 7, 12), (8, 'Libreria Bogotá', 8, 11),

(9, 'Libreria Medellin', 1, 11), (10, 'Libreria Medellin', 2, 12), (11, 'Libreria Medellin', 3, 14),

(12, 'Libreria Medellin', 4, 7), (13, 'Libreria Medellin', 5, 9), (14, 'Libreria Medellin', 6, 9),

(15, 'Libreria Medellin', 7, 4), (16, 'Libreria Medellin', 8, 6);

**Para el Nodo 2: base de datos “librería\_sur”**

create table sucursal\_libro(

codigo int not null,

nombre\_sucursal varchar(50) not null,

codigo\_libro int not null,

cantidad\_ejemplares int not null,

primary key(codigo),

foreign key(nombre\_sucursal) references sucursal(nombre),

foreign key(codigo\_libro) references libro(codigo)

);

insert into sucursal\_libro VALUES (17, 'Libreria Pasto', 1, 11), (18, 'Libreria Pasto', 2, 8),

(19, 'Libreria Pasto', 3, 12), (20, 'Libreria Pasto', 4, 8), (21, 'Libreria Pasto', 5, 8),

(22, 'Libreria Pasto', 6, 4), (23, 'Libreria Pasto', 7, 12), (24, 'Libreria Pasto', 8, 11),

(25, 'Libreria Cali', 1, 11), (26, 'Libreria Cali', 2, 12), (27, 'Libreria Cali', 3, 14),

(28, 'Libreria Cali', 4, 7), (29, 'Libreria Cali', 5, 9), (30, 'Libreria Cali', 6, 9),

(31, 'Libreria Cali', 7, 4), (32, 'Libreria Cali', 8, 6);

**Para el Nodo 3: base de datos “eps\_sur”**

create table sucursal\_libro(

codigo int not null,

nombre\_sucursal varchar(50) not null,

codigo\_libro int not null,

cantidad\_ejemplares int not null,

primary key(codigo),

foreign key(nombre\_sucursal) references sucursal(nombre),

foreign key(codigo\_libro) references libro(codigo)

);

insert into sucursal\_libro VALUES (40, 'Libreria Santa Marta', 1, 7), (33, 'Libreria Santa Marta', 2, 3),

(34, 'Libreria Santa Marta', 3, 2), (35, 'Libreria Santa Marta', 4, 6),

(36, 'Libreria Santa Marta', 5, 6), (37, 'Libreria Santa Marta', 6, 4),

(38, 'Libreria Santa Marta', 7, 12), (39, 'Libreria Santa Marta', 8, 11),

(41, 'Libreria Riohacha', 1, 15), (42, 'Libreria Riohacha', 2, 7), (43, 'Libreria Riohacha', 3, 7),

(44, 'Libreria Riohacha', 4, 15), (45, 'Libreria Riohacha', 5, 23), (46, 'Libreria Riohacha', 6, 21),

(47, 'Libreria Riohacha', 7, 9), (48, 'Libreria Riohacha', 8, 11), (49, 'Libreria Barranquilla', 1, 23),

(51, 'Libreria Barranquilla', 2, 9), (52, 'Libreria Barranquilla', 3, 8),

(53, 'Libreria Barranquilla', 4, 7), (54, 'Libreria Barranquilla', 5, 7),

(55, 'Libreria Barranquilla', 6, 4), (56, 'Libreria Barranquilla', 7, 15),

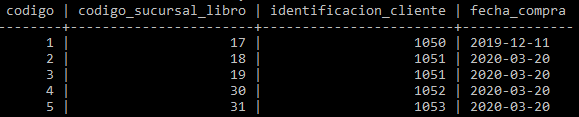
(57, 'Libreria Barranquilla', 8, 15), (58, 'Libreria Cartagena', 1, 14),

(59, 'Libreria Cartagena', 2, 14), (61, 'Libreria Cartagena', 3, 12), (62, 'Libreria Cartagena', 4, 8),

(63, 'Libreria Cartagena', 5, 12), (64, 'Libreria Cartagena', 6, 11), (65, 'Libreria Cartagena', 7, 16),

(66, 'Libreria Cartagena', 8, 16);

**Venta:** La tabla *venta* guarda los datos de las ventas por cada sucursal. Consta de cuatro columnas: código, código\_sucursal\_libro, identificación\_cliente y fecha\_compra como lo muestra la siguiente imagen, resultado de ejecutar la sentencia *select \* from venta:*

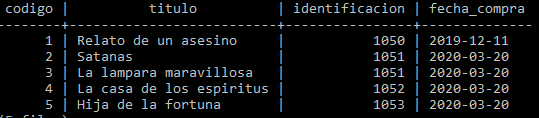


Para observar más claramente la tabla *venta*, se puede realizar una reunión natural con las tablas venta, sucursal\_libro y libro, de tal forma que se muestre en nombre del libro en vez de código\_sucursal\_libro de tal forma que se muestre que libros a comprado cada cliente:

select v.codigo, titulo, c.identificacion, fecha\_compra

from venta v join sucursal\_libro sl on sl.codigo = v.codigo\_sucursal\_libro join libro l on l.codigo = sl.codigo\_libro join cliente c on c.identificacion = v.identificacion\_cliente;

Cuyo resultado es el siguiente:



**Justificación de la fragmentación:** Debido a que la tabla *venta* se fragmentó, como se explicó anteriormente y que en la tabla *sucursal\_libro* existe una llave foránea (código\_sucursal\_libro) que hace referencia al atributo codigo de la tabla *sucursal\_libro*, es obligatorio que la tabla *sucursal\_libro* también sea fragmentada.

Por tanto, la tabla *venta* se puede fragmentar horizontalmente de la tal manera que las tuplas se fragmenten de acuerdo a los libros vendidos que está registrado en la tabla (codigo) y, cada uno de esos fragmentos se coloquen en cada uno de los nodos dependiendo de en cual nodo esté dicho venta.

Debido a que se realiza una fragmentación horizontal, el esquema de la base de datos para cada nodo es el mismo:

|  |  |  |
| --- | --- | --- |
| **Nodo1** | **Nodo2** | **Nodo3** |
|  |  |  |

.

El DDL y el DML de la base de datos quedan de la siguiente manera:

**Para el Nodo 1: base de datos “libreria\_centro”**

create table venta(

codigo int not null,

codigo\_sucursal\_libro int not null,

identificacion\_cliente int not null,

fecha\_compra date not null DEFAULT CURRENT\_DATE,

primary key(codigo),

foreign key(codigo\_sucursal\_libro) references sucursal\_libro(codigo),

foreign key(identificacion\_cliente) references cliente(identificacion)

);

insert into venta VALUES

(1, 1, 1044, '2019-12-12');

insert into venta VALUES (2, 2, 1044), (3, 3, 1045), (4, 9, 1048), (5, 12, 1048), (6, 11, 1046);

insert into venta VALUES

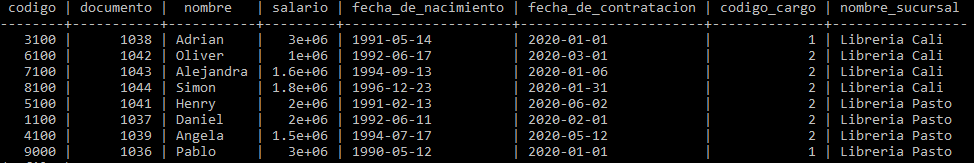
(7, 13, 1047, '2019-12-24');

insert into venta values (8, 1, 1046);

insert into venta values (9, 14, 1045);

insert into venta values (10, 12, 1049);

**Empleado:** La tabla *empleado* guarda los datos correspondientes de cada empleado vincualado a cada sucursal. Consta solo con ocho columnas: código, documento, nombre, salario, fecha\_nacimiento, fecha\_contratacion, código\_cargo y nombre\_sucursal, como lo muestra la siguiente imagen, resultado de ejecutar la sentencia *select \* from empleado order by nombre\_sucursal:*



**Justificación de la fragmentación:** Puede ser de interés mantener la información de los empleados vinculados de forma separada teniendo en cuenta la ubicación geográfica de las sucursales a las cuales están registrados. Teniendo en cuenta esto, si un empleado está vinculado en la sucursal de Cali sus datos deberían estar en el nodo 2 y no en los otros nodos, ya que el nodo 2, agrupa los datos de las sucursales Cali y Pasto.

Por tanto, la tabla *empleado* se puede fragmentar horizontalmente de la tal manera que las tuplas se fragmenten de acuerdo a la sucursal en la cual está contratado el empleado y, cada uno de esos fragmentos se coloque en cada uno de los nodos, como ya se indicó anteriormente.

Debido a que se realiza una fragmentación horizontal, el esquema de la base de datos para cada nodo es el mismo:

|  |  |  |
| --- | --- | --- |
| **Nodo1** | **Nodo2** | **Nodo3** |
|  |  |  |

.

El DDL y el DML de la tabla queda de la siguiente manera:

**Para el Nodo 2: base de datos “eps\_sur”**

create table empleado(

codigo int not null,

documento int not null unique,

nombre varchar(30) not null,

salario real not null,

fecha\_de\_nacimiento date not null,

fecha\_de\_contratacion date not null,

codigo\_cargo int not null,

nombre\_sucursal varchar(40) not null,

foreign key(nombre\_sucursal) references sucursal(nombre),

primary key(codigo)

);

insert into empleado values

(9000, 1036, 'Pablo', 3000000, '1990-05-12', '2020-01-01', 1, 'Libreria Pasto'),

(1100, 1037, 'Daniel', 2000000, '1992-06-11', '2020-02-01', 2, 'Libreria Pasto'),

(3100, 1038, 'Adrian', 3000000, '1991-05-14', '2020-01-01', 1, 'Libreria Cali'),

(4100, 1039, 'Angela', 1500000, '1994-07-17', '2020-05-12', 2, 'Libreria Pasto'),

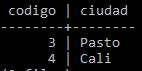
(5100, 1041, 'Henry', 2000000, '1991-02-13', '2020-06-02', 2, 'Libreria Pasto'),

(6100, 1042, 'Oliver', 1000000, '1992-06-17', '2020-03-01', 2, 'Libreria Cali'),

(7100, 1043, 'Alejandra', 1600000, '1994-09-13', '2020-01-06', 2, 'Libreria Cali'),

(8100, 1044, 'Simon', 1800000, '1996-12-23', '2020-01-31', 2, 'Libreria Cali');

**Ciudad:** La tabla *ciudad* guarda los datos de las ciudades en donde están ubicadas las sucursales. Consta solamente de dos columnas: código y ciudad, como lo muestra la siguiente imagen, resultado de ejecutar la sentencia *select \* from ciudad:*



**Justificación de la fragmentación:** Puede ser de interés mantener la información de las sucursales de forma separada teniendo en cuenta la ubicación geográfica de la ciudad a la que pertenece. Por ejemplo, si una sucursal se encuentra ubicada en la ciudad de Barranquilla, sus datos deberían estar en el nodo 3 y no en los otros nodos, ya que el nodo 3, como vimos anteriormente, agrupa los datos de las sucursales Santa Marta, Barranquilla, Cartagena y Riohacha.

Por tanto, la tabla *ciudad* se puede fragmentar horizontalmente de la tal manera que las tuplas se fragmenten de acuerdo a la ciudad a la cual pertenece una sucursal y, cada uno de esos fragmentos se coloque en cada uno de los nodos, como ya se indicó anteriormente.

Debido a que se realiza una fragmentación horizontal, el esquema de la base de datos para cada nodo es el mismo:

|  |  |  |
| --- | --- | --- |
| **Nodo1** | **Nodo2** | **Nodo3** |
|  |  |  |

El DDL y el DML de la tabla queda de la siguiente manera:

**Para el Nodo 2: base de datos “eps\_sur”**

create table ciudad(

codigo int not null,

ciudad varchar(25) not null,

primary key(codigo)

);

insert into ciudad values (3, 'Pasto'), (4, 'Cali');

1. **Definición de consultas**

**NOTA:** La ejecución de las consultas se hará desde el nodo 1 (el que tiene la base de datos “libreria\_centro”). Es importante tener esto en cuenta, ya que así se ve reflejado en los scripts de las consultas a continuación. Las consultas se encuentran en el archivo “consultas”:

1. Consolidado mensual de la cantidad de clientes nuevos a nivel nacional.
2. Consolidado mensual y anual de la editorial de la cual se han vendido más libros en cada sucursal.
3. Datos de los clientes nuevos. Estos datos organizados por sucursal en un mes particular.
4. Libros más comprados en un mes particular a nivel nacional por cada sucursal.
5. Datos de los empleados a nivel nacional organizados por sucursal y por cargo de tal forma que sea posible identificar el empleado con mayor y menor antigüedad.