ITERACION 3-PROYECTO DEL CURSO

Juan David Cardona Páez, Nicolás Quintero Acevedo

Iteración 3 Proyecto AforoCC-Andes

Universidad de los Andes, Bogotá, Colombia

{jd.cardonap, n.quinteroa}@uniandes.edu.co

Fecha de presentación: Noviembre 16 de 2020

1. Introducción.

El objetivo de este texto es enseñar la implementación de AFORO-CCANDES, un servicio integrado de monitoreo y control del aforo de un centro comercial en general y de los establecimientos comerciales que tiene. El objetivo principal es hacer respetar la norma de aislamiento social y también contener la velocidad de un eventual contagio mediante 11 requerimientos funcionales y 9 requerimientos funcionales de consulta

2. Creación Esquema AforoCC-Andes.

Para crear cada una de las tablas de AforoCC-Andes en SQL se usó el siguiente script, el cual sigue el orden de cada una de las figuras (1-10) y el modelo presentado en la Iteracion 1 para asegurarse de la correcta creación de las tablas y las respectivas relaciones establecidas dentro de estas. Es importante resaltar el último constraint de la tabla CARNET, ya que esta sentencia solo puede ser declarada después de crear la tabla VISITANTE para crear las referencias.

```
CREATE TABLE CENTRO_COMERCIAL
    (
     NOMBRE VARCHAR2(255 BYTE),
     AFORO FLOAT NOT NULL,
     CONSTRAINT centro_comercial_pk PRIMARY KEY(NOMBRE));
```

Figura 1. Script Creación tabla CENTRO_COMERCIAL

```
CREATE TABLE ESPACIO
(
  ID_ESPACIO NUMBER NOT NULL,
  HORARIO_APERTURA_EMPLEADOS TIMESTAMP NOT NULL,
  HORARIO_APERTURA_CLIENTES TIMESTAMP NOT NULL,
  HORARIO_CIERRE_CLIENTES TIMESTAMP NOT NULL,
  AFORO_ACTUAL INT NOT NULL,
  AFORO_TOTAL INT NOT NULL,
  ESTADO VARCHAR2(255 BYTE),
  CONSTRAINT espacio_pk PRIMARY KEY (ID_ESPACIO));

ALTER TABLE ESPACIO
ADD CONSTRAINT ck_estado
CHECK(ESTADO IN ('Desocupado','Verde','Deshabilitado','Rojo','Naranja'))
ENABLE;
```

Figura 2. Script Creación tabla ESPACIO

```
CREATE TABLE LOCAL_COMERCIAL
(
  IDESPACIO NUMBER  NOT NULL,
  ID_LOCAL NUMBER NOT NULL,
  NOMBRE VARCHAR2(255 BYTE) NOT NULL,
  NOMBRE_EMPRESA VARCHAR2(255 BYTE) NOT NULL,
  AREA FLOAT NOT NULL,
  TIPO_ESTABLECIMIENTO VARCHAR2(255 BYTE) NOT NULL,
  CONSTRAINT local_comercialpk PRIMARY KEY(ID_LOCAL)) ;
ALTER TABLE LOCAL_COMERCIAL
ADD CONSTRAINT fk_local
    FOREIGN KEY (IDESPACIO)
    REFERENCES espacio(ID_ESPACIO)
ENABLE;
```

Figura 3. Script Creación tabla LOCAL_COMERCIAL

```
CREATE TABLE PARQUEADERO(
IDESPACIO NUMBER NOT NULL,
ID_PARQUEADERO NUMBER NOT NULL,
CAPACIDAD FLOAT NOT NULL,
CONSTRAINT parqueadero_pk PRIMARY KEY(ID_PARQUEADERO)
);
ALTER TABLE PARQUEADERO
ADD CONSTRAINT fk_parqueadero
  FOREIGN KEY (IDESPACIO)
  REFERENCES espacio(ID_ESPACIO)
ENABLE;
```

Figura 4. Script Creación tabla PARQUEADERO

```
CREATE TABLE BAÑO(
IDESPACIO NUMBER NOT NULL,
ID_BAÑO NUMBER NOT NULL,
NUMERO_SANITARIOS INT NOT NULL,
CONSTRAINT baño_pk PRIMARY KEY (ID_BAÑO));

  ALTER TABLE BAÑO
ADD CONSTRAINT fk_baño
  FOREIGN KEY (IDESPACIO)
  REFERENCES espacio(ID_ESPACIO)
ENABLE;
CREATE TABLE LECTOR
```

Figura 5. Script Creación tabla BAÑO

```
CREATE TABLE LECTOR
(
  ID_LECTOR NUMBER NOT NULL,
  IDESPACIO NUMBER NOT NULL,
  CONSTRAINT lector_pk PRIMARY KEY(ID_LECTOR));
ALTER TABLE LECTOR
ADD CONSTRAINT fk_espacio
    FOREIGN KEY (IDESPACIO)
    REFERENCES espacio(ID_ESPACIO)
ENABLE;
```

Figura 6. Script Creación tabla LECTOR

```
CREATE TABLE CARNET
(
  ID_CARNET NUMBER NOT NULL,
  CEDULA FLOAT NOT NULL,
  CONSTRAINT carnet_pk PRIMARY KEY(ID_CARNET));
```

Figura 7. Script Creación tabla CARNET

```
CREATE TABLE VISITA
(
  FECHAYHORA_OP TIMESTAMP NOT NULL,
  TIPO_OP VARCHAR2(255 BYTE) NOT NULL,
  HORAFIN_OP TIMESTAMP NOT NULL,
  IDLECTOR NUMBER NOT NULL,
  IDCARNET NUMBER NOT NULL,
  IDESPACIO NUMBER NOT NULL,
  CONSTRAINT visita_pk PRIMARY KEY(IDLECTOR, IDCARNET, IDESPACIO,TIPO_OP,FECHAYHORA_OP));

ALTER TABLE VISITA
ADD CONSTRAINT fk_visita_lector
    FOREIGN KEY(IDLECTOR)
    REFERENCES LECTOR(ID_LECTOR)
ENABLE;

ALTER TABLE VISITA
ADD CONSTRAINT fk_visita_carnet
    FOREIGN KEY(IDCARNET)
    REFERENCES carnet(ID_CARNET)
ENABLE;

ALTER TABLE VISITA
ADD CONSTRAINT fk_visita_espacio
    FOREIGN KEY(IDESPACIO)
    REFERENCES espacio(ID_ESPACIO)
ENABLE;
```

Figura 8. Script Creación tabla VISITA

```
ENABLE;
```

```
CREATE TABLE VISITANTE
( CEDULA FLOAT NOT NULL,
  NOMBRE VARCHAR2(255 BYTE) NOT NULL,
  TELEFONO FLOAT NOT NULL,
  NOMBRE_CONTACTO VARCHAR2(255 BYTE) NOT NULL,
  TELEFONO_CONTACTO FLOAT NOT NULL,
  CODIGO_QR VARCHAR2(255 BYTE) NOT NULL,
  CORREO VARCHAR2(255 BYTE) NOT NULL,
  HORARIO_DISPONIBLE TIMESTAMP NOT NULL,
  TIPO_VISITANTE VARCHAR2(255 BYTE) NOT NULL,
  IDESPACIO NUMBER NOT NULL,
  ESTADO VARCHAR2(255 BYTE) NOT NULL,
  CONSTRAINT visitante_pk PRIMARY KEY(CEDULA));

ALTER TABLE VISITANTE
ADD CONSTRAINT ck_tipo_visitante
    CHECK(TIPO_VISITANTE IN ('Empleado','Vigilancia','Aseo','Mantenimiento','Clientes','Domiciliarios','Administrador_centro','Administrador_local
ENABLE;

ALTER TABLE VISITANTE
ADD CONSTRAINT ck_estado_visitante
    CHECK(ESTADO IN ('Positivo', 'Rojo', 'Naranja','Verde'))
ENABLE;

ALTER TABLE VISITANTE
ADD CONSTRAINT fk_espacio_visitante
    FOREIGN KEY(IDESPACIO)
    REFERENCES espacio(ID_ESPACIO)
```

```
ENABLE;
```

Figura 9. Script Creación tabla VISITANTE

```
  ALTER TABLE CARNET
  ADD CONSTRAINT fk_cedula
  FOREIGN KEY (CEDULA)
  REFERENCES   visitante(CEDULA)
  ENABLE;
COMMIT;
```

Figura 10. Script Constraint asignación llave foránea tabla CARNET

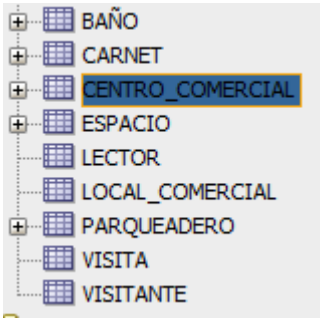

Figura 11. Tablas creadas AforoCC-Andes

1FN: No deben existen atributos multivalor (dominios atómicos para cada atributo). En los modelos, tanto conceptual como relacional se evidencia que no ninguna clase ni entidad posee un atributo multivalor como un array, una lista, un arreglo, etc. Por lo tanto, este nivel de normalización se superó.

• 2FN: Está en 1FN y no hay dependencias parciales desde los atributos primos (los atributos no primos dependen de forma completa de todas las llaves candidatas). Los atributos que no son llaves en nuestro modelo conceptual y relacional solo se pueden conocer por medio de una llave primaria o foránea, pero nunca a raíz de otro atributo no primo. Por tanto, este nivel de normalización se supera.

• 3FN: Está en 2FN y no existen dependencias transitivas entre atributos no primos. Como se dijo en 2NF, no se puede acceder a un atributo no primo a raíz de otro no primo. Es decir, no hay dependencias parciales entre ninguna tupla de atributos no primos, por ende, tampoco hay dependencias transitivas entre atributos no primos. En todas las dependencias funcionales la

parte izquierda es una llave o la parte derecha es prima. Este nivel es superado por el modelo establecido.

• BCFN (Boyce-Codd): Sí está en 3NF y las llaves son simples, entonces es BCFN. Ya estábamos en el nivel 3 y ninguna de las llaves establecidas son compuestas. Solo hay llaves simples para poder acceder a los atributos no primos en el modelo conceptual como lo son idCarnet, cedula, idLector y idEspacio. **Por eso mismo, el modelo se encuentra dentro de este nivel de normalización.**

3. Requerimientos funcionales de modificación.

El requerimiento funcional 1 es registrar cada uno de los espacios del centro comercial por el administrador del centro comercial. Sabiendo esto, se genera una secuencia SQL donde se llenan de datos las columnas de ESPACIO, PARQUEADERO, LOCAL_COMERCIAL Y BAÑO en el orden respectivo. Hay que tener en cuenta que PARQUEADERO, LOCAL_COMERCIAL y BAÑO, no se pueden crear si no tienen un ESPACIO al cual pueden ser asignados.

```
insert into ESPACIO (ID_ESPACIO, HORARIO_APERTURA_EMPLEADOS, HORARIO_APERTURA_CLIENTES, HORARIO_CIERRE_CLIENTES, AFORO_ACTUAL, AFORO_TOTAL,ESTADO)
values (1,to_date('2020-12-14:07:00:00', 'YYYY-MM-DD:HH24:MI:SS'),to_date('2020-12-14:10:00:00', 'YYYY-MM-DD:HH24:MI:SS'),
to_date('2020-12-14:18:00:00', 'YYYY-MM-DD:HH24:MI:SS'),8, 90,'Verde');
```

Figura 12. Sentencia SQL para registrar un ESPACIO

```
insert into PARQUEADERO (ID_PARQUEADERO,IDESPACIO,  CAPACIDAD) values (1,36,43);
insert into PARQUEADERO (ID_PARQUEADERO,IDESPACIO,  CAPACIDAD) values (2,37, 43);
insert into PARQUEADERO (ID_PARQUEADERO,IDESPACIO,  CAPACIDAD) values (3,38, 33);
insert into PARQUEADERO (ID_PARQUEADERO,IDESPACIO,  CAPACIDAD) values (4,39,26);
insert into PARQUEADERO (ID_PARQUEADERO,IDESPACIO,  CAPACIDAD) values (5,40,15);
```

Figura 13. Sentencia SQL para registrar un PARQUEADERO

```
insert into LOCAL_COMERCIAL (ID_LOCAL,IDESPACIO, NOMBRE, NOMBRE_EMPRESA, AREA, TIPO_ESTABLECIMIENTO)
values (1,11, 'Hayes LLC', 'Demivee', 222.87, 'Restaurante');
insert into LOCAL_COMERCIAL (ID_LOCAL,IDESPACIO, NOMBRE, NOMBRE_EMPRESA, AREA, TIPO_ESTABLECIMIENTO)
values (2,12, 'Franecki, Hills and Pfeffer', 'Divape', 244.53, 'Tienda de ropa');
insert into LOCAL_COMERCIAL (ID_LOCAL,IDESPACIO,  NOMBRE, NOMBRE_EMPRESA, AREA, TIPO_ESTABLECIMIENTO)
values (3,13, 'Bartell Group', 'Yotz', 275.24, 'Farmacia');
```

Figura 14. Sentencia SQL para registrar un LOCAL_COMERCIAL

```
insert into BANO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (1,41, 10);
insert into BAÑO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (2,42, 5);
insert into BAÑO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (3,43, 7);
insert into BAÑO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (4,44, 6);
insert into BAÑO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (5,45, 11);
insert into BAÑO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (6,46, 12);
insert into BAÑO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (7,47, 13);
insert into BAÑO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (8,48, 20);
insert into BAÑO (ID_BAÑO,IDESPACIO,  NUMERO_SANITARIOS) values (9,49, 2);
insert into BAÑO (ID_BAÑO, IDESPACIO, NUMERO_SANITARIOS) values (10,50, 15);
```

Figura 15. Sentencia SQL para registrar un BAÑO

El requerimiento funcional 2 consta de como registrar los establecimientos al centro comercial y como se puede observar en la figura 14, se crea un LOCAL_COMERCIAL y se asigna a su respectivo espacio.

Para el requerimiento funcional 3 y 4 que son respectivamente, registrar los tipos de visitante en el centro comercial y registrar un visitante al centro comercial se usa la sentencia SQL de inserción para la tabla VISITANTE como se observa en la figura 16. Además, en la figura 9 se observa como con la sentencia CHECK se registran los tipos de visitante del centro comercial y el estado de este mismo.

```
insert into VISITANTE (CEDULA, NOMBRE, TELEFONO, NOMBRE_CONTACTO, TELEFONO_CONTACTO, CODIGO_QR, CORREO, HORARIO_DISPONIBLE, IDESPACIO,TIPO_VISITANTE,ESTA
values ('6304894842726358991', 'Garrik Olufsen', '2232486935', 'Boone Cromett', '2707144449', 'http://dummyimage.com/131x181.jpg/cc0000/ffffff',
'bcromett0@europa.eu', to_date('2020-12-14:8:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 22,'Empleado','Verde');
```

Figura 16. Sentencia SQL para registrar un visitante y tipos de visitante en la tabla VISITANTE.

Para el requerimiento número 5, el cual es registrar los lectores del carnet, se hace una inserción con la sentencia SQL INSERT en la tabla con la información del Espacio al cual pertenece como se observa en la figura 17, pues este nos ayudará a determinar más adelante si el lector pertenece al centro comercial o al establecimiento.

```
insert into LECTOR (ID_LECTOR, IDESPACIO) values (1, 1);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (2, 2);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (3, 3);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (4, 4);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (5, 5);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (6, 6);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (7, 7);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (8, 8);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (9, 9);
insert into LECTOR (ID_LECTOR, IDESPACIO) values (10, 10);
```

Figura 17. Sentencia SQL para registrar un lector en la tabla LECTOR.

El requerimiento funcional 6 es registrar la entrada o salida de un visitante a uno de los espacios del centro comercial. Para esto, se hace uso de la sentencia SQL INSERT en la tabla VISITA para registrar el respectivo tipo de operación(entrada/salida) con la fecha inicio de la operación y referenciando el lector donde se hace la visita, el espacio y el carnet del visitante que hace la visita.

```
insert into VISITA(FECHAYHORA_OP,TIPO_OP,HORAFIN_OP,IDLECTOR,IDCARNET,IDESPACIO)
values(to_date('2020-12-14:8:00:00', 'YYYY-MM-DD:HH24:MI:SS'),'Entrada',to_date('2020-12-14:14:00:00', 'YYYY-MM-DD:HH24:MI:SS'),22,1,22);
insert into VISITA(FECHAYHORA_OP,TIPO_OP,HORAFIN_OP,IDLECTOR,IDCARNET,IDESPACIO)
values(to_date('2020-12-14:23:00:00', 'YYYY-MM-DD:HH24:MI:SS'),'Entrada',to_date('2020-12-15:00:00:00', 'YYYY-MM-DD:HH24:MI:SS'),36,2,36);
insert into VISITA(FECHAYHORA_OP,TIPO_OP,HORAFIN_OP,IDLECTOR,IDCARNET,IDESPACIO)
values(to_date('2020-12-14:08:00:00', 'YYYY-MM-DD:HH24:MI:SS'),'Entrada',to_date('2020-12-14:08:30:00', 'YYYY-MM-DD:HH24:MI:SS'),48,3,48);
```

Figura 18. Sentencia SQL para registrar la entrada o salida de un visitante en la tabla VISITA.

Para el requerimiento funcional 7 de cerrar un establecimiento del centro comercial se hace uso de la sentencia SQL DELETE en la tabla LOCAL_COMERCIAL en, la cual deja el espacio disponible para lo que desee hacer el administrador del centro comercial.

```
DELETE FROM LOCAL_COMERCIAL WHERE ID_LOCAL= 1;
DELETE FROM LOCAL_COMERCIAL WHERE NOMBRE = 'Nike';
```

Figura 19. Sentencia SQL para cerrar un establecimiento en la tabla LOCAL_COMERCIAL.

4. Requerimientos funcionales de Consulta.

Para el requerimiento funcional de consulta 1, el cual es mostrar todos los visitantes atendidos por un establecimiento se hace una sentencia SQL SELECT FROM donde se encuentran cada uno de los visitantes de un establecimiento con ayuda del elemento BETWEEN para encontrar el rango de las fechas e INNER JOIN para moverse entre las relaciones necesarias.

```
--REQUERIMIENTO DE CONSULTA 1
--Para administradores de establecimientos comerciales
SELECT Carnet, VISITANTE.CEDULA, VISITANTE.NOMBRE, VISITANTE.TELEFONO, VISITANTE.NOMBRE_CONTACTO, VISITANTE.TELEFONO_CONTACTO, VISITANTE.CORREO
FROM(SELECT IDCARNET as Carnet
    FROM VISITA
    WHERE idespacio=? AND FECHAYHORA_OP BETWEEN ? AND ?)
    INNER JOIN CARNET ON CARNET.ID_CARNET=Carnet
    INNER JOIN VISITANTE ON CARNET.CEDULA=VISITANTE.CEDULA;
--Para el administrador del centro
SELECT Carnet, xs as IdEspacio, VISITANTE.CEDULA, VISITANTE.NOMBRE, VISITANTE.TELEFONO, VISITANTE.NOMBRE_CONTACTO, VISITANTE.TELEFONO_CONTACTO, VISITA
FROM (SELECT IDCARNET as Carnet, IDESPACIO as xs
    FROM VISITA
    WHERE FECHAYHORA_OP BETWEEN ? AND ?)
    INNER JOIN CARNET ON CARNET.ID_CARNET=Carnet
    INNER JOIN VISITANTE ON CARNET.CEDULA=VISITANTE.CEDULA;
```

Figura 20. Sentencia SQL para mostrar todos los visitantes atendidos por un establecimiento.

Para el requerimiento funcional de consulta 2, el cual es mostrar todos los 20 establecimientos más populares se hace una sentencia SQL SELECT FROM donde se encuentran cada uno de los establecimientos con ayuda del elemento BETWEEN para encontrar el rango de las fechas e INNER JOIN para moverse entre las relaciones necesarias, HAVING COUNT para encontrar que mínimo alguien ha visitado el establecimiento y ORDER BY el contador de los establecimientos para organizar la lista de forma descendente y con WHERE ROWNUM<= 20 nos aseguramos de mostrar solo 20 establecimientos más populares.

```
SELECT LOCAL_COMERCIAL.idespacio as IdEstablecimiento,local_comercial.nombre, local_comercial.tipo_establecimiento as Tipo,contadorVisitas
FROM(SELECT IDESPACIO as IDESPACIOXD, COUNT(DISTINCT IDCARNET)as contadorVisitas
    FROM VISITA
    WHERE FECHAYHORA_OP BETWEEN ? AND ?
    GROUP BY IDESPACIO
    HAVING COUNT(DISTINCT IDCARNET)>0)
    INNER JOIN LOCAL_COMERCIAL ON IDESPACIOXD=LOCAL_COMERCIAL.idespacio
    WHERE ROWNUM<=20
    ORDER BY contadorVisitas DESC;
```

Figura 21. Sentencia SQL para mostrar todos los 20 establecimientos más populares.

Para el requerimiento funcional de consulta 3, el cual es mostrar el índice del aforo del centro comercial se hace uso de la sentencia SQL SELECT, FROM y WHERE, para retornar el índice de aforo del centro comercial. Por otro lado, cuando se trata de un establecimiento se hace uso de las mismas sentencias, pero con otros dos SELECT FROM anidados en el que mediante el uso de la sentencia WHERE solo se consulta el índice de su propio establecimiento. También, es importante resaltar el uso de INNER JOIN donde se encuentra la información mediante la relación entre ESPACIO y LOCAL_COMERCIAL.

```
--REQUERIMIENTO DE CONSULTA 3
--PARA EL INDICE DE AFORO DEL CC DEL ADMIN DEL ESPACIO
SELECT AforoEnCC/AforoMaximo as IndiceAforoEnCC, IdEspacio, AforoEnEstablecimiento/AforoMaximoXd as AforoEnEspacio
FROM(SELECT SUM(aforo_actual) as AforoEnCC, SUM(aforo_total) as AforoMaximo
    FROM ESPACIO),
    (SELECT ID_ESPACIO as IdEspacio, aforo_actual as AforoEnEstablecimiento, aforo_total as AforoMaximoXd
    FROM ESPACIO
    WHERE ID_ESPACIO=?);
--PARA EL INDICE DE AFORO DEL CC PARA ADIMINISTRADOR DEL CENTRO
--Segun el establecimiento
SELECT AforoEnCC/AforoMaximo as IndiceAforoEnCC, IdEspacio, AforoEnEstablecimiento/AforoMaximoXd as AforoEnEspacio
FROM(SELECT SUM(aforo_actual) as AforoEnCC, SUM(aforo_total) as AforoMaximo
    FROM ESPACIO),
    (SELECT ID_ESPACIO as IdEspacio, aforo_actual as AforoEnEstablecimiento, aforo_total as AforoMaximoXd
    FROM ESPACIO
    WHERE ID_ESPACIO=?);
--Segun el tipo de establecimiento
SELECT AforoEnCC/AforoMaximo as IndiceAforoEnCC,TipoEspacio,  AforoActualEspacio/AforoTotalEspacio as IndiceTipoEstablecimiento
FROM(SELECT SUM(aforo_actual) as AforoEnCC, SUM(aforo_total) as AforoMaximo
    FROM ESPACIO),
    (SELECT TipoEspacio, SUM(espacio.aforo_actual) as AforoActualEspacio, SUM(espacio.aforo_total) as AforoTotalEspacio
    FROM(SELECT IDESPACIO as IdEspacio, TIPO_ESTABLECIMIENTO as TipoEspacio
        FROM LOCAL_COMERCIAL
        WHERE TIPO_ESTABLECIMIENTO=?)
        INNER JOIN ESPACIO ON espacio.id_espacio=IdEspacio group by TipoEspacio);
```

Figura 22. Sentencia SQL para mostrar el índice de aforo del centro comercial.

Para el requerimiento funcional de consulta 4, el cual es mostrar los establecimientos con aforo disponible del centro comercial se hace uso de la sentencia SQL SELECT, FROM y WHERE, para retornar los establecimientos con aforo disponible con la condición AFORO_ACTUAL<AFORO_TOTAL.

```
--RFC4
SELECT ID_ESPACIO, AFORO_ACTUAL, AFORO_TOTAL, ESTADO
FROM ESPACIO
WHERE AFORO_ACTUAL<AFORO_TOTAL;
```

Figura 23. Sentencia SQL para mostrar los establecimientos con aforo disponible.

Para el requerimiento funcional de consulta 5, el cual es mostrar el comportamiento de un tipo de visitante dentro del centro comercial se hace uso de la sentencia SQL SELECT, FROM y WHERE, para conocer todos los visitantes de ese tipo, gracias a la condición TIPO_VISITATE=?. Después, se hace un INNER JOIN con CARNET para identificar el IdCarnet de todos los visitantes y así, después, hacer un INNER JOIN con VISITA donde identificamos las cédulas de todos los visitantes de ese tipo. Así, después solo es extraer la información del visitante y de las visitas que ha realizado entre el intervalo de tiempo establecida. Por otro lado, cuando se trata de un tipo de local se hace uso de las mismas sentencias, pero en vez de empezar por visitante se empieza por LOCAL_COMERCIAL para conocer el Id del espacio y el nombre del local. Con esto, después se hace un INNER JOIN con visita donde identificamos las visitas realizadas a ese espacio en un tiempo establecido. Por último, usamos un GROUP BY de los Id de los espacios para poder contar el número de visitas por medio de la columna TIPO_OP, y se retorna la información pertinente.

```
--RFC5 por TIPO DE VISITANTE
SELECT CedulaVis, nombre, VISITA.FECHAYHORA_OP, VISITA.TIPO_OP, VISITA.HORAFIN_OP, visita.idespacio
FROM (SELECT CEDULA as CedulaVis, NOMBRE as nombre
      FROM VISITANTE
      WHERE TIPO_VISITANTE=?)
INNER JOIN CARNET ON carnet.cedula=CedulaVis
INNER JOIN VISITA ON CARNET.ID_CARNET=IDCARNET
WHERE visita.fechayhora_op BETWEEN ? AND ?;
--RFC5 por TIPO DE LOCAL
SELECT IdEspacioXd, nombre, COUNT(VISITA.TIPO_OP) as NumeroVisitas
FROM (SELECT IDESPACIO as IdEspacioXd, NOMBRE as nombre
      FROM LOCAL_COMERCIAL
      WHERE TIPO_ESTABLECIMIENTO=?)
INNER JOIN VISITA ON IdEspacioXd=visita.idespacio
WHERE visita.fechayhora_op BETWEEN ? AND ?
GROUP BY IdEspacioXd, nombre;
```

Figura 24. Sentencia SQL para mostrar el comportamiento de un visitante o tipo de local.

Para el requerimiento funcional de consulta 6, el cual quiere mostrar el comportamiento de un visitante en específico dentro del centro comercial, se hace uso de la sentencia SQL SELECT, FROM y WHERE en la tabla VISITANTE para conocer la cedula y nombre del visitante buscado, gracias a la condición CEDULA=?. Después hacemos un INNER JOIN con CARNET para saber el id del carnet del usuario y después in INNER JOIN con VISITA para tener información de las visitas que realizó. Con la información de las visitas, se hace otro INNER JOIN con LOCAL_COMERCIAL para saber qué establecimientos comerciales visitó en un intervalo de tiempo específico. Por último, se extrae la información relevante.

```
--RFC6
SELECT cedulaXd, nombreOP, visita.fechayhora_op, visita.tipo_op, visita.horafin_op, visita.idespacio, LOCAL_COMERCIAL.NOMBRE
FROM (SELECT CEDULA as cedulaXd, NOMBRE as nombreOP
      FROM VISITANTE
      WHERE CEDULA=?)
INNER JOIN CARNET ON carnet.cedula=cedulaXd
INNER JOIN VISITA ON carnet.id_carnet=VISITA.IDCARNET
INNER JOIN LOCAL_COMERCIAL ON VISITA.IDESPACIO=LOCAL_COMERCIAL.IDESPACIO
WHERE visita.fechayhora_op BETWEEN ? AND ?;
```

Figura 25. Sentencia SQL para mostrar el comportamiento de un visitante.

5. Nuevos requerimientos funcionales de Consulta y modificación.

Para el requerimiento funcional de consulta 7, el cual es mostrar las horas de mayor afluencia, de mayor y menor riesgo del centro comercial en un intervalo de tiempo específico según una medida de tiempo y un tipo de establecimiento, se dividió este procedimiento en 3 partes y después se unieron los resultados con el comando UNION.

1era parte: se hace uso de la sentencia SQL SELECT, FROM y WHERE nombrada como A, para retornar todas las visitas en el intervalo de la medida de tiempo establecida y se agrupan principalmente por el id del espacio de las visitas para así poder contar cuántas visitas tuvo cada espacio en ese intervalo de tiempo. Después, se hace un INNER JOIN con LOCAL_COMERCIAL para filtrar los espacios por el tipo de establecimiento estipulado por el usuario. Además, en el INNER JOIN obtenemos datos como el nombre del establecimiento. Luego, agrupamos todos los datos por fechas para así poder sumar, para cada fecha, cuál fue la suma de las visitas. Después, ordenamos por la suma de visitas de cada fecha de manera descendente para que el comando WHERE ROWNUM<2 escoja a la fecha que más visitas obtuvo.

Además de esta sentencia, se hace otra aparte nombrada como B para obtener la suma del aforo total de todos los establecimientos de ese tipo.

```
--RFC7
--Fecha Mayor afluencia
(SELECT A.flecho as flechin,B.SumaTotalAforoMaximo
FROM(SELECT flecho, jjjj, establi, idEsXd, SUM(numVisitas) as Sumavisitas
FROM(SELECT idEsXd, flecho, numVisitas, LOCAL_COMERCIAL.NOMBRE as jjjj, LOCAL_COMERCIAL.TIPO_ESTABLECIMIENTO as establi
FROM(SELECT IDESPACIO as idEsXd, FECHAYHORA_OP as flecho, COUNT(TIPO_OP) as NumVisitas
FROM VISITA
WHERE FECHAYHORA_OP BETWEEN to_date('2020-12-14:14:00:00', 'YYYY-MM-DD:HH24:MI:SS') AND to_date('2020-12-14:19:00:00', 'YYYY-MM-DD:HH24:MI:SS')
GROUP BY IDESPACIO, FECHAYHORA_OP)
INNER JOIN LOCAL_COMERCIAL ON LOCAL_COMERCIAL.IDESPACIO=idEsXd AND LOCAL_COMERCIAL.TIPO_ESTABLECIMIENTO='Restaurante')
GROUP BY flecho,  jjjj, establi, idEsXd
ORDER BY SUM(numVisitas) DESC)A,
(SELECT SUM(cd) as SumaTotalAforoMaximo
FROM(SELECT ID_ESPACIO as idd, AFORO_TOTAL as cd
FROM ESPACIO)
INNER JOIN LOCAL_COMERCIAL ON LOCAL_COMERCIAL.IDESPACIO=idd AND local_comercial.tipo_establecimiento='Restaurante') B
WHERE ROWNUM<2)UNION
```

Figura 26. Sentencia SQL para mostrar las horas de mayor afluencia.

2da parte: se hace uso de la sentencia SQL SELECT, FROM y WHERE nombrada como A, para retornar todas las visitas en el intervalo de la medida de tiempo establecida y se agrupan principalmente por el id del espacio de las visitas para así poder contar cuántas visitas tuvo cada espacio en ese intervalo de tiempo. Después, se hace un INNER JOIN con LOCAL_COMERCIAL para filtrar los espacios por el tipo de establecimiento estipulado por el usuario. Además, en el INNER JOIN obtenemos datos como el nombre del establecimiento. Luego, agrupamos todos los datos por fechas para así poder sumar, para cada fecha, cuál fue la suma de las visitas. Después, ordenamos por la suma de visitas de cada fecha de manera descendente.

Además de esta sentencia, se hace otra aparte nombrada como B para obtener el 90% la suma del aforo total de todos los establecimientos de ese tipo. De esta manera, cogemos el atributo de la sentencia A llamado SumaVisitas2 para comparar si es mayor al el 90% la suma del aforo total de los establecimientos de ese tipo, ya que esa es la condición para que una fecha se considere de alto riesgo. Debido a nuestras inserciones en la base de datos, esta parte de la consulta hasta el momento no retorna nada porque ninguna fecha cumple tal condición. Por último, se hace el ROMNUM<2 para obtener la primera fecha del registro.

```
(SELECT A.flecho2 as flechin2, b.noventaporcientoaforomaximo as XD2
FROM(SELECT flecho2, jjjj2, establi2, idEsXd2, SUM(numVisitas2) as Sumavisitas2
FROM(SELECT idEsXd2, flecho2, numVisitas2, LOCAL_COMERCIAL.NOMBRE as jjjj2, LOCAL_COMERCIAL.TIPO_ESTABLECIMIENTO as establi2
FROM(SELECT IDESPACIO as idEsXd2, FECHAYHORA_OP as flecho2, COUNT(TIPO_OP) as NumVisitas2
FROM VISITA
WHERE FECHAYHORA_OP BETWEEN to_date('2020-12-14:14:00:00', 'YYYY-MM-DD:HH24:MI:SS') AND to_date('2020-12-14:19:00:00', 'YYYY-MM-DD:HH24:MI:SS')
GROUP BY IDESPACIO, FECHAYHORA_OP)
INNER JOIN LOCAL_COMERCIAL ON LOCAL_COMERCIAL.IDESPACIO=idEsXd2 AND LOCAL_COMERCIAL.TIPO_ESTABLECIMIENTO='Restaurante')
GROUP BY flecho2,  jjjj2, establi2, idEsXd2
ORDER BY SUM(numVisitas2) DESC)A,
(SELECT SUM(cd)*0.9 as NoventaPorcientoAforoMaximo
FROM(SELECT ID_ESPACIO as idd, AFORO_TOTAL as cd
FROM ESPACIO)
INNER JOIN LOCAL_COMERCIAL ON LOCAL_COMERCIAL.IDESPACIO=idd AND local_comercial.tipo_establecimiento='Restaurante') B
WHERE A.SumaVisitas2>b.noventaporcientoaforomaximo AND ROWNUM<2 UNION(
```

Figura 27. Sentencia SQL para mostrar las horas de mayor riesgo.

3ra parte: se hace uso de la sentencia SQL SELECT, FROM y WHERE nombrada como A, para retornar todas las visitas en el intervalo de la medida de tiempo establecida y se agrupan principalmente por el id del espacio de las visitas para así poder contar cuántas visitas tuvo cada espacio en ese intervalo de tiempo. Después, se hace un INNER JOIN con LOCAL_COMERCIAL para filtrar los espacios por el tipo de establecimiento estipulado por el usuario. Además, en el INNER JOIN obtenemos datos como el nombre del establecimiento. Luego, agrupamos todos los datos por

fechas para así poder sumar, para cada fecha, cuál fue la suma de las visitas. Después, ordenamos por la suma de visitas de cada fecha de manera ascendente (se busca la fecha con mínimo riesgo).

Además de esta sentencia, se hace otra aparte nombrada como B para obtener el 10% la suma del aforo total de todos los establecimientos de ese tipo. De esta manera, cogemos el atributo de la sentencia A llamado SumaVisitas3 para comparar si es menor al el 10% la suma del aforo total de los establecimientos de ese tipo, ya que esa es la condición para que una fecha se considere de menor riesgo. Por último, se hace el ROMNUM<2 para obtener la primera fecha del registro (la de menor riesgo).

```
SELECT A.flecho3 as flechin3, b.diezporcientoaforomaximo as XD3
FROM(SELECT flecho3, jjjj3, establi3, idEsXd3, SUM(numVisitas3) as Sumavisitas3
FROM(SELECT idEsXd3, flecho3, numVisitas3, LOCAL_COMERCIAL.NOMBRE as jjjj3, LOCAL_COMERCIAL.TIPO_ESTABLECIMIENTO as establi3
FROM(SELECT IDESPACIO as idEsXd3, FECHAYHORA_OP as flecho3, COUNT(TIPO_OP) as NumVisitas3
FROM VISITA
WHERE FECHAYHORA_OP BETWEEN to_date('2020-12-14:14:00:00', 'YYYY-MM-DD:HH24:MI:SS') AND to_date('2020-12-14:19:00:00', 'YYYY-MM-DD:HH24:MI:SS')
GROUP BY IDESPACIO, FECHAYHORA_OP)
INNER JOIN LOCAL_COMERCIAL ON LOCAL_COMERCIAL.IDESPACIO=idEsXd3 AND LOCAL_COMERCIAL.TIPO_ESTABLECIMIENTO='Restaurante')
GROUP BY flecho3,  jjjj3, establi3, idEsXd3
ORDER BY SUM(numVisitas3) ASC)A,
(SELECT SUM(cd)*0.1 as DiezPorcientoAforoMaximo
FROM(SELECT ID_ESPACIO as idd, AFORO_TOTAL as cd
FROM ESPACIO)
INNER JOIN LOCAL_COMERCIAL ON LOCAL_COMERCIAL.IDESPACIO=idd AND local_comercial.tipo_establecimiento='Restaurante') B
WHERE A.SumaVisitas3<b.diezporcientoaforomaximo AND ROWNUM<2);
```

Figura 28. Sentencia SQL para mostrar las horas de menor riesgo.

Por último, se hace UNION entre las tres sentencias y se extraen las fechas junto a un valor int que representa: (1) para la fecha de mayor afluencia, la suma de los aforos máximos de cada establecimiento de ese tipo; (2) para la fecha de mayor riesgo, el 90% de la suma de los aforos máximos de cada establecimiento de ese tipo; (3) para la fecha de menor riesgo, el 10% de la suma de los aforos máximos de cada establecimiento de ese tipo.

Para el requerimiento funcional de consulta 8, el cual quiere mostrar los clientes frecuentes de un establecimiento dado, se hace uso de la sentencia SQL SELECT, FROM y WHERE, para retornar el id del espacio y el nombre del establecimiento. Después, se hace un INNER JOIN con VISITA para obtener las visitas hechas a ese espacio y obtener la información del mes de la operación (EXTRACT (MONTH FROM CAST (FECHAYHORA_OP as DATE))) porque es necesaria para saber cuántas veces en un mes ha ido un visitante al establecimiento. Asimismo, se obtiene el número de visitas en el mes a ese establecimiento hecha por un usuario del cual se extrae su Id. Toda esta última información se agrupa por el Id del carnet del visitante, precisamente para que el contador de visitas del usuario al establecimiento por mes sea preciso. Después se filtran los visitantes donde sus visitas en el mes son mayores o iguales a 3. Aparte, se hacen dos INNER JOIN con Carnet y con Visitante para obtener la información del visitante y eliminar de la lista a los que son de mantenimiento o Domiciliarios. Por último, se agrupa por el nombre del visitante frecuente.

```
--RFC8
SELECT VISITANTE.NOMBRE, VISITANTE.TELEFONO, VISITANTE.CORREO, idCarnetUsuario, numVisitasEseMes, mes
FROM(SELECT aidi, name, mes, idCarnetUsuario, numVisitasEseMes
FROM(SELECT aidi, name, EXTRACT(MONTH FROM CAST(FECHAYHORA_OP as DATE)) as mes, VISITA.IDCARNET as idCarnetUsuario, COUNT(VISITA.FECHAYHORA_OP) as numVisitasEseMes
FROM(SELECT IDESPACIO as aidi, NOMBRE as name
FROM LOCAL_COMERCIAL
WHERE NOMBRE='Hayes LLC')
INNER JOIN VISITA ON VISITA.IDESPACIO=aidi
GROUP BY VISITA.IDCARNET, EXTRACT(MONTH FROM CAST(FECHAYHORA_OP as DATE)), aidi, name)
WHERE numVisitasEseMes>=3)
INNER JOIN CARNET ON CARNET.ID_CARNET=idCarnetUsuario
INNER JOIN VISITANTE ON CARNET.CEDULA=VISITANTE.CEDULA AND NOT VISITANTE.TIPO_VISITANTE='Mantenimiento' AND NOT VISITANTE.TIPO_VISITANTE='Domiciliarios'
GROUP BY VISITANTE.NOMBRE, VISITANTE.TELEFONO, VISITANTE.CORREO, idCarnetUsuario, numVisitasEseMes, mes;
```

Figura 29. Sentencia SQL para mostrar el índice de aforo del centro comercial.

Para el requerimiento funcional de consulta 9, el cual es mostrar los visitantes que tuvieron contacto entre sí dada una fecha y sus 10 días anteriores, se hace uso de la sentencia SQL SELECT, FROM y WHERE, para retornar todas las visitas que fueron en el mismo lugar y a la misma hora entre la fecha establecida y sus 10 días anteriores. Después, se hace un INNER JOIN con carnet y visitante para obtener la información de los visitantes y, después, se hace un GROUP BY para filtrar por fecha y por carnet los visitantes que estuvieron implicados en el mismo espacio a la misma hora y se ordena por el id del espacio de manera ascendente.

```
--RFC9
SELECT fecha, usuarioCarnet, VISITANTE.NOMBRE, VISITANTE.TELEFONO, VISITANTE.CORREO, op, espacio
FROM(SELECT visitaxs.IDESPACIO espacio, visitaxs.FECHAYHORA_OP fecha, visitaxs.IDCARNET as usuarioCarnet, visitaxs.tipo_op as op
FROM VISITA visitaxs, (SELECT * FROM VISITA) visita2
WHERE visitaxs.IDESPACIO=visita2.IDESPACIO
AND visitaxs.idlector=visita2.IDLECTOR
AND visitaxs.fechayhora_op=visita2.fechayhora_op
AND NOT visitaxs.idcarnet=visita2.idcarnet
AND visitaxs.fechayhora_op BETWEEN ?-10 AND ?)
INNER JOIN CARNET ON CARNET.ID_CARNET=usuarioCarnet
INNER JOIN VISITANTE ON CARNET.CEDULA=VISITANTE.CEDULA
GROUP BY fecha,usuarioCarnet,VISITANTE.NOMBRE, VISITANTE.TELEFONO, VISITANTE.CORREO, op, espacio
ORDER BY espacio ASC;
```

Figura 30. Sentencia SQL para mostrar el índice de aforo del centro comercial.

Para el requerimiento funcional 8 se hace una sentencia UPDATE en la cual se inserta el ESTADO nuevo del VISITANTE con SET basado en su CEDULA con la sentencia WHERE.

```
-Requerimiento Funcional 8
UPDATE VISITANTE
SET ESTADO='?'
WHERE CEDULA = '?';
```

Figura 31. Sentencia SQL para registrar el cambio de estado de un visitante.

Para el requerimiento funcional 9 se hace una sentencia UPDATE en la cual se inserta el Estado nuevo del ESPACIO con SET basado en su ID_ESPACIO con la sentencia WHERE.

```
-Requerimiento Funcional 9
UPDATE ESPACIO
SET ESTADO ='?'
WHERE ID_ESPACIO='?';
```

Figura 32. Sentencia SQL para registrar el cambio de espacio de un visitante.

Para el requerimiento funcional 11 se usa la sentencia UPDATE para el ESTADO y se cambia la columna con SET y la columna ESTADO pasa a ser Deshabilitado y AFORO_ACTUAL pasa a ser igual 0 y con la sentencia WHERE para asegurarme de cambiar el espacio que yo quiero sin id. Además, de la misma forma se usa un UPDATE de la tabla VISITA para finalizar el proceso de ENTRADA de los visitantes en ese ESPACIO con la ID_ESPACIO para así desalojar los visitantes del espacio cambiando la columna HORAFIN_OP con SET por el valor GETDATE() que retorna la fecha actual y con el WHERE se cerciora de que la FECHAYHORA_OP sea inferior a la actual para asegurarnos de que estamos desalojando visitantes que no hayan salido.

```sql
--Requerimiento Funcional 11
UPDATE ESPACIO
SET ESTADO= 'Deshabilitado',
    AFORO_ACTUAL= 0
WHERE ID_ESPACIO='?';
UPDATE VISITA
SET HORAFIN_OP = GETDATE()
WHERE IDESPACIO ='?' AND FECHAYHORA_OP< CONVERT(DATE,GETDATE()) AND TIPO_OP='Entrada';
```

Figura 33. Sentencia SQL para deshabilitar el cambio de espacio de un visitante.

Para el requerimiento funcional 12 se usa la sentencia UPDATE para actualizar la tabla ESPACIO y modificar su columna ESTADO con el valor ‘Verde’ para indicar que está habilitada nuevamente. La sentencia WHERE se usa para que basado en la ID_ESPACIO se haga la respectiva actualización.

```sql
--Requerimiento Funcional 12
UPDATE ESPACIO
SET ESTADO='Verde'
WHERE ID_ESPACIO='?';
```

Figura 34. Sentencia SQL para habilitar el ESPACIO nuevamente.

6. Bibliografía.

1. D. Ullman, J., Widom, J., & Garcia-Molina, H. (2008). *Database Systems The Complete Book*. Prentice Hall.