ITERACION 2-PROYECTO DEL CURSO

Juan David Cardona Páez, Nicolás Quintero Acevedo

Iteración 1 Proyecto AforoCC-Andes

Universidad de los Andes, Bogotá, Colombia

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

Fecha de presentación: Octubre 19 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 7 requerimientos funcionales y 3 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,
  CONSTRAINT espacio_pk PRIMARY KEY (ID_ESPACIO));
```

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));

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

```
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,
  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 fk_espacio_visitante
    FOREIGN KEY(IDESPACIO)
```

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

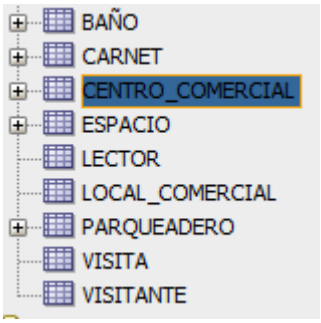

Figura 11. Tablas creadas AforoCC-Andes

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)
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);
```

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.

```
insert into VISITANTE (CEDULA, NOMBRE, TELEFONO, NOMBRE_CONTACTO, TELEFONO_CONTACTO, CODIGO_QR, CORREO, HORARIO_DISPONIBLE, IDESPACIO,TIPO_VISITANTE
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');
```

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.

```
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;
```

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

```
--Para el administrador del centro
SELECT Carnet, xs as IdEspacio, VISITANTE.CEDULA, VISITANTE.NOMBRE, VISITANTE.TELEFONO, VISITANTE.NOMBRE_CONTACTO, VISITANTE.TELEFONO_CONTACTO,
VISITANTE.CORREO
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;
```

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.

Figura 21. Sentencia SQL para mostrar todos los visitantes atendidos por un establecimiento para el administrador.

```
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 22. 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.

```sql
--REQUERIMIENTO DE CONSULTA 3
--PARA EL INDICE DE AFORO DEL CC
SELECT *
FROM(SELECT SUM(aforo_actual) as AforoEnCC
    FROM ESPACIO),
    (SELECT aforo_actual as AforoEnEstablecimiento
    FROM ESPACIO
    WHERE ID_ESPACIO=?);
--PARA EL INDICE DE AFORO DEL CC PARA ADIMINISTRADOR DEL CENTRO
--Segun el establecimiento
SELECT *
FROM(SELECT SUM(aforo_actual) as AforoEnCC
    FROM ESPACIO),
    (SELECT aforo_actual as AforoEnEstablecimiento
    FROM ESPACIO
    WHERE ID_ESPACIO=?);
--Segun el tipo de establecimiento
SELECT AforoEnCC, TipoEspacio, AforoEnEspacio
FROM(SELECT SUM(aforo_actual) as AforoEnCC
    FROM ESPACIO),
    (SELECT espacio.aforo_actual as AforoEnEspacio, TipoEspacio
    FROM(SELECT IDESPACIO as IdEspacio, TIPO_ESTABLECIMIENTO as TipoEspacio
        FROM LOCAL_COMERCIAL
        WHERE TIPO_ESTABLECIMIENTO=?)
        INNER JOIN ESPACIO ON espacio.id_espacio=IdEspacio);
-----
```

Figura 23. Sentencia SQL para mostrar el índice de aforo del centro comercial.

5. Bibliografía.

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