# ESCUELA COLOMBIANA DE INGENIERÍA

**MODELOS Y BASES DE DATOS - MBDA G01 – 2020-1 PREPARCIAL**

**SEGUNDO TERCIO**

**Nombres: Ricardo Olarte, Andres Calderon**

# ALIMENTAR

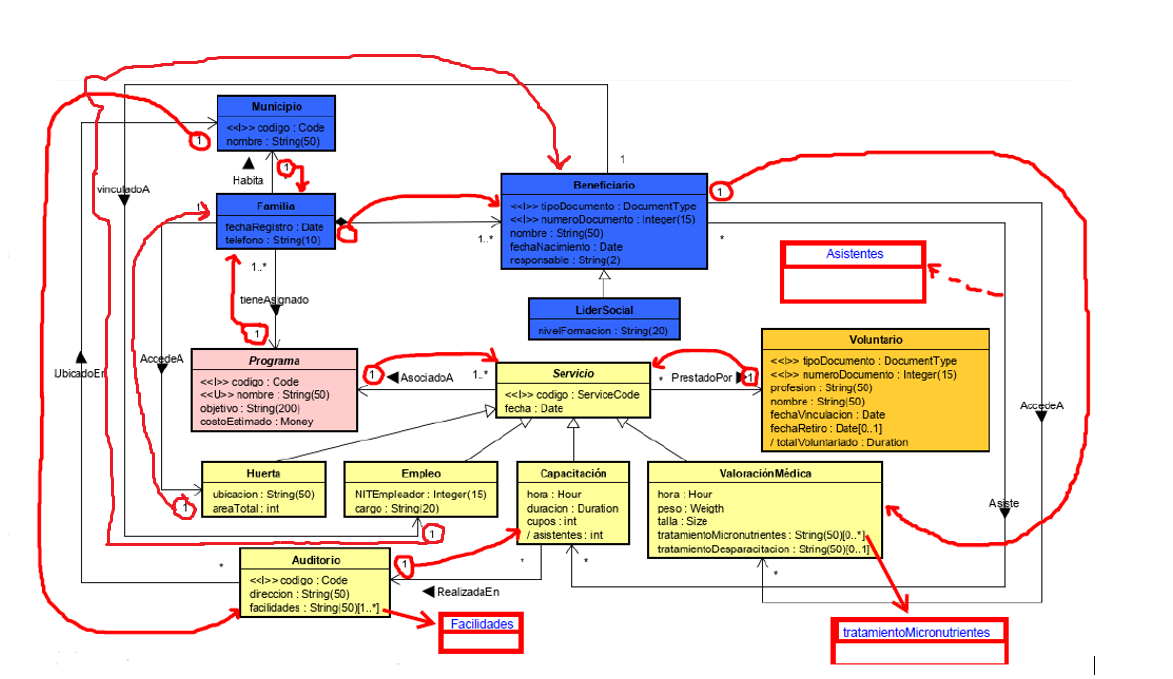
**En este ciclo de trabajo vamos a implementar el área de Servicios.**

**SERVICIOS**

ALIMENTAR presta los programas de la siguiente manera:

En las valoraciones médicas se realizan controles de peso y talla a todos los miembros de la familia y tratamientos de micronutrientes o desparasitación, si algún miembro lo requiere.

En cuanto a las capacitaciones, los representantes de cada familia o los líder social pueden recibir cualquiera de los programas de capacitación ofrecidas. De las capacitaciones ofrecidas es importante conocer el programa de capacitación, fecha, hora, duración, lugar, municipio, capacitador, número de cupos y asistentes. En cada municipio hay auditorios, autorizados por ALIMENTAR, para realizar los programas de capacitación; de ellos interesa nombre, dirección y facilidades que ofrece (cafetería, grabaciones, etc). El programa de acceso a alimentos es implementado por un equipo de donantes voluntarios, quienes realizan el enlace entre la familia y empresas (en caso del acceso económico) o ayudan a la familia a preparar su huerta ( en caso del acceso directo). En todos los casos es importante conocer el detalle del servicio prestado, incluyendo el tiempo invertido por cada uno de los donantes que participaron en el servicio.



## (30%)MODELO RELACIONAL (Modelo mini)

Transforme el modelo conceptual extendido anterior en un modelo relacional usando el estándar mínimo. No olvide presentar el diseño de su solución sobre el diagrama conceptual. (ubicación de relaciones y tablas)

Una captura de pantalla de una red social

Descripción generada automáticamente

## (20%) MODELO RELACIONAL. INTEGRIDAD DECLARATIVA. (SQL Estándar)

Seleccione e implemente los mecanismos necesarios para implementar las siguientes restricciones:

1. El tipo de documento puede ser: ‘CC’ o ‘TI’
   * CREATE DOMAIN DocumentType AS VARCHAR(2) ADD CONSTRAINT CHECK(VALUE IN ('CC','TI'));
2. Cada capacitación tiene un cupo máximo de 20 asistentes.
   * ALTER TABLE Capacitaciones ADD CONSTRAINT CHK\_Capacitacion\_Cupos CHECK(cupo<=20 AND cupo>=0);
3. Un beneficiario no puede asistir a más de 5 capacitaciones.
   * CREATE ASSERTION as\_capacitacionBeneficiario CHECK(NOT EXIST(SELECT COUNT(\*)

FROM ASISTENTES

GROUP BY BeneficiariotipoDocumento,BeneficiarionumeroDocumento,BeneficiariofamiliaMunicipio,BeneficiariofamiliaPrograma,BeneficiariofamiliafechaRegistro,Beneficiariofamiliatelefono,Beneficiariofamiliahuerta

HAVING COUNT(\*)>5));

1. Si un voluntario se ha retirado, la fecha de retiro no puede ser anterior a la fecha de vinculación ni posterior a la fecha actual.
   * ALTER TABLE Voluntarios ADD CONSTRAINT CHK\_FechaRetiro CHECK(fechaRetiro>=fechaVinculacion AND fechaRetrio<=SYSDATE());
2. La herencia de los servicios es completa.

* CREATE ASSERTION HerenciaCompleta CHECK(NOT EXIST(SELECT Codigo FROM Servicio NOT IN ((SELECT Codigo FROM Huerta) UNION (SELECT Codigo FROM Empresa) UNION (SELECT Codigo FROM Capacitacion) UNION (SELECT Codigo FROM ValoracionMedica)));

## (20%) MODELO RELACIONAL. INTEGRIDAD PROCEDIMENTAL. (SQL Estándar)

Seleccione y explique los mecanismos necesarios para cumplir con los siguientes reglas de negocio: **(SOLO DISEÑO. NO LAS IMPLEMENTE)**

## Registrar Servicio

## Adicionar

* + La adición se debe realizar con datos mínimos. (Indique los datos a automatizar)
    - CREATE TRIGGER BEFORE INSERT; para poder calcular los datos automáticamente
      * Servicio: Fecha
      * Capacitacion: Hora
      * ValoracionMedica: Hora
  + El código de cada servicio está definido por el prefijo SRx, donde x es un número de 5 dígitos.
    - CREATE TRIGGER ### BEFORE INSERT
    - Disparador antes de insertar, que si no se cumple con el prefijo lance una error
    - Esto se hace para poder corroborar la integridad del atributo ‘codigo’ antes de que este sea adicionado a la base de datos
  + No se puede adicionar un servicio relacionado con un programa diferente al que tiene asignado cada familia.
    - CREATE TRIGGER ### BEFORE INSERT
    - Disparador antes de insertar, que verifique que el servicio se relaciones con la familia dada
    - La razón de este disparador es que verifique el servicio antes de insertarlo
  + No se puede asignar un servicio a un voluntario que se encuentre retirado.
    - CREATE TRIGGER ### BEFORE INSERT
    - Disparador antes de insertar que verifique que el voluntario nuevo no este retirado(su fecha de retiro sea diferente a null)
* Modificar
  + No se puede cambiar el programa asociado a cada servicio.
    - CREATE TRIGGER ### INSTEAD OF UPDATE
    - Disparador INSTEAD OF udate, para poder bloquear las actualización donde se quiera modificar el programa de un servicio
  + El voluntario asignado a un servicio se puede cambiar siempre y cuando no se haya retirado.
    - CREATE TRIGGER ### BEFORE UPDATE
    - Disparador antes de udate, para poder corroborar que cuando se vaya a modificar el voluntario este no se haya retirado y en caso de que si se haya retirado dejar el voluntario anterior
  + Si se modifica el tipo y número de documento de un beneficiario, se actualizan los servicios asociados a él.
    - CREATE TRIGGER ### AFTER UPDATE
    - Disparador después de actualización, para después de actualizar el beneficiario actualizar los servicios a los cuales este está asociado
  + De cada tipo de servicio se puede modificar:
    - Huerta: Área total.
      * CREATE TRIGGER ### BEFORE UPDATE
      * Disparador antes de Update, para solo hacer los cambios al atributo Area total
    - Empleo: Cargo.
      * CREATE TRIGGER ### BEFORE UPDATE
      * Disparador antes de Update, para solo hacer los cambios al atributo Cargo
    - Capacitación: Hora y auditorio, siempre y cuando la fecha del servicio sea anterior a la fecha actual.
      * CREATE TRIGGER ### BEFORE UPDATE
      * Disparador antes de Update, para solo verificar que se cumpla que la fecha del servicio sea anterior a la fecha actual; y si esto pasa solamente modificar los atributos Hora y auditorio; y los demás dejarlos con los valores old
    - Valoración médica: Hora siempre y cuando la fecha del servicio sea anterior a la fecha actual.
      * CREATE TRIGGER ### BEFORE UPDATE
      * Disparador antes de Update, para solo verificar que se cumpla que la fecha del servicio sea anterior a la fecha actual; y si esto pasa solamente modificar el atributo Hora y los demás dejarlos con los valores old
* Eliminar
  + Un servicio no se puede eliminar si la fecha es posterior a la fecha actual.
    - CREATE TRIGGER BEFORE DELETE
    - Para verificar que se cumpla con la fecha; y en caso de que no se cumpla lanzar un mensaje de error
  + Si se elimina un beneficiario, los servicios asociados a él quedan referenciando a un beneficiario indefinido.
    - CONSTRAINT ### FOREIGN KEY ### REFERENCES ### ON DELETE SET NULL
    - Para cada vez que se elimine la pk a la cual se hace referencia este espacio sea puesto con valor null

## (30%) CONSTRUCCIÓN (SQL ORACLE)

1. Implemente la estructura de las tablas diseñadas en el punto 1. (sin restricciones externas).
   * CREATE TABLE Beneficiarios(

tipoDocumento VARCHAR(2) NOT NULL,

numeroDocumento NUMBER(15) NOT NULL,

nombre VARCHAR(50) NOT NULL,

fechaNacimiento DATE NOT NULL,

responsable VARCHAR(2) NOT NULL

familiaMunicipio NUMBER(10) NOT NULL,

familiaPrograma NUMBER(10) NOT NULL,

familiafechaRegistro DATE NOT NULL,

familiatelefono VARCHAR(10) NOT NULL,

familiahuerta NUMBER(10) NOT NULL,

empleo NUMBER(10) NOT NULL);

* CREATE TABLE LideresSociales(

BeneficiariotipoDocumento VARCHAR(2) NOT NULL,

BeneficiarionumeroDocumento NUMBER(15) NOT NULL,

BeneficiariofamiliaMunicipio NUMBER(10) NOT NULL,

BeneficiariofamiliaPrograma NUMBER(10) NOT NULL,

BeneficiariofamiliafechaRegistro DATE NOT NULL,

Beneficiariofamiliatelefono VARCHAR(10) NOT NULL,

Beneficiariofamiliahuerta NUMBER(10) NOT NULL,

nivelFormacion VARCHAR(20) NOT NULL);

* CREATE TABLE Municipios(

codigo NUMBER(10) NOT NULL,

nombre VARCHAR(50) NOT NULL);

* CREATE TABLE Familias(

fechaRegistro DATE NOT NULL,

telefono VARCHAR(10) NOT NULL,

municipio NUMBER(10) NOT NULL,

programa NUMBER(10) NOT NULL,

huerta NUMBER(10) NOT NULL);

* CREATE TABLE Programas(

codigo NUMBER(10) NOT NULL,

nombre VARCHAR(50) NOT NULL,

objetivo VARCHAR(200) NOT NULL,

costoEstimado NUMBER(10) NOT NULL);

* CREATE TABLE Servicios(

codigo NUMBER(10) NOT NULL,

fecha DATE NOT NULL,

voluntariotipoDocumento NUMBER(10) NOT NULL,

voluntarionumeroDocumento NUMBER(15) NOT NULL);

* CREATE TABLE Huertas(

codigo NUMBER(10) NOT NULL,

ubicacion VARCHAR(50) NOT NULL,

areaTotal NUMBER(10) NOT NULL);

* CREATE TABLE Empleos(

codigo NUMBER(10) NOT NULL,

NITEmpleador NUMBER(15) NOT NULL,

cargo VARCHAR(20) NOT NULL);

* CREATE TABLE Capacitaciones(

codigo NUMBER(10) NOT NULL,

hora NUMBER(4) NOT NULL,

duracion NUMBER(10) NOT NULL,

cupos NUMBER(2) NOT NULL);

* CREATE TABLE ValoracionesMedicas

codigo NUMBER(10) NOT NULL,

hora NUMBER(4) NOT NULL,

peso NUMBER(3) NOT NULL,

talla VARCHAR(4) NOT NULL,

tratamientoDesparacitacion VARCHAR(50));

* CREATE TABLE Voluntarios(

tipoDocumento NUMBER(10) NOT NULL,

numeroDocumento NUMBER(15) NOT NULL,

profesion VARCHAR(50) NOT NULL,

nombre VARCHAR(50) NOT NULL,

fechaVinculacion DATE NOT NULL,

fechaRetiro DATE);

* CREATE TABLE Auditorios(

codigo NUMBER(10) NOT NULL,

direccion VARCHAR(50) NOT NULL,

municipio NUMBER(10) NOT NULL);

* CREATE TABLE Facilidades(

auditorio NUMBER(10) NOT NULL,

facilidades VARCHAR(50) NOT NULL);

* CREATE TABLE tratamientosMicronutrientes(

ValoracionMedica NUMBER(10) NOT NULL,

tratamientosMicronutrientes VARCHAR(50) );

* CREATE TABLE Asistentes(

BeneficiariotipoDocumento VARCHAR(2) NOT NULL,

BeneficiarionumeroDocumento NUMBER(15) NOT NULL,

BeneficiariofamiliaMunicipio NUMBER(10) NOT NULL,

BeneficiariofamiliaPrograma NUMBER(10) NOT NULL,

BeneficiariofamiliafechaRegistro DATE NOT NULL,

Beneficiariofamiliatelefono VARCHAR(10) NOT NULL,

Beneficiariofamiliahuerta NUMBER(10) NOT NULL,

capacitacion NUMBER(10) NOT NULL);

1. Implemente las restricciones externas declarativas correspondientes al CRUD Servicio
   * ALTER TABLE Capacitaciones ADD CONSTRAINT CHK\_hora CHECK(MOD(hora,100)<60 AND hora<=2359);
   * ALTER TABLE ValoracionesMedicas ADD CONSTRAINT CHK\_talla CHECK(talla IN (‘XS’, ’S’, ’M’, ’L’, ‘XL’, ‘XXL’, ‘XXXL’));
   * ALTER TABLE ValoracionesMedicas ADD CONSTRAINT CHK\_peso CHECK( peso>3 AND peso<300);
   * ALTER TABLE ValoracionesMedicas ADD CONSTRAINT CHK\_hora CHECK(MOD(hora,100)<60 AND hora<=2359);
2. Implemente los mecanismos diseñados para cumplir las reglas del caso de uso. Registrar Servicio.

## BONO (+.2)

Se puede hacer una mejora al diseño conceptual anterior. ¿Cuál propondría?

* Suavizaríamos las relaciones 1 a 1
  + Familia – Huerta ---> suavizaríamos huerta
  + Beneficiario – Empleo ---> suavizaríamos empleo
* Crearíamos un nuevo concepto llamado PersonaNatural para que este sea el concepto mas general y que los conceptos voluntarios y beneficiarios fueran conceptos heredados de este
  + Que tenga los siguientes valores:
    - tipoDocumento
    - numeroDocumento
    - nombre

# Sintaxis

|  |  |
| --- | --- |
| **Tipo** | **CREATE DOMAIN** ……. **AS** …….. **CONSTRAINT CHECK** ……….. |
| **Atributo y tupla** | **ALTER TABLE** ……… **ADD CONSTRAINT** …….. **CHECK** ……….. |
| **Tabla y base de datos** | **CREATE ASSERTION** ………… **CHECK** ………… |
| **Acciones** | **CONSTRAINT** *…………* **FOREIGN KEY** ……… **REFERENCES** …….  **[ON DELETE (RESTRICT** | **CASCADE** | **SET NULL)**] **[ON UPDATE (RESTRICT** | **CASCADE** | **SET NULL)**] |
| **Disparadores** | **CREATE TRIGGER** nombre  **[BEFORE | AFTER |INSTEAD OF ]**  **[DELETE|INSERT|UPDATE [OF** nombre columnas**]]**  **ON** nombre Tabla  **[FOR EACH ROW [WHEN (**condición**)]]**  **DECLARE** -- Definición de variables  **BEGIN**  -- Instrucciones del disparador  **END** nombre;  En la tercer línea puede incluirse más de una acción uniéndose con un OR |