

# Bases de Datos 2

*Obligatorio*

---

Carrera: ATI / AP  
Grupo: M3A

Estudiantes: Guillermo Polachek (153924) - Sebastián Villar (177751)

Docente: Fernando Martínez

Año 2017

# Contenido

<b>Contenido</b>	<b>2</b>
<b>Descripción de la solución</b>	<b>3</b>
Creación de tablas	3
Restricciones	4
Índices	4
Inserción de datos de prueba	4
Procedimientos y funciones	5
Disparadores	5
Consultas SQL	5
Vistas	6
<b>Script con las restricciones de integridad creadas sobre el script de creación de tablas, índices, y el ingreso de datos de prueba</b>	<b>7</b>
<b>Script con la creación de disparadores</b>	<b>24</b>
<b>Script con la creación de funciones y procedimientos</b>	<b>31</b>
<b>Script con la resolución de las consultas y vista</b>	<b>40</b>
<b>Anexo con juego de datos a rechazar</b>	<b>46</b>
<b>Impresión de los scripts</b>	<b>52</b>
Scripts de restricciones de integridad	52
Restricciones para UNIVERSIDAD	52
Restricciones para INVESTIGADOR	52
Restricciones para TRABAJO	53
Restricciones para TAGS / TTAGS	54
Restricciones para TAUtores	55
Restricciones para REFERENCIAS	55
Restricciones para LUGARES	56
Scripts creación Índices	58
Scripts de ingreso de datos de prueba	58
Script con la creación de funciones y procedimientos	64
Script con la creación de disparadores	72
Script con la resolución de las consultas y vista	78

## Descripción de la solución

### Creación de tablas

La base de datos se generó a partir del script de creación de tablas que aparece en la letra del obligatorio.

Sobre este script, en particular, corresponde comentar lo siguiente:

- En la tabla Trabajo debería existir un atributo fechaFin para asignar una fecha de finalización a aquellos trabajos terminados. Sin esta fecha final no es posible determinar cuando un trabajo fue culminado o no.
- También en la tabla Trabajo debería agregarse un atributo fechaPublicado para asignar una fecha de publicación a cada trabajo. Si esta fecha es nula, el trabajo aún no ha sido publicado. Una ventaja de agregar este atributo es que podría utilizarse en una condición de combinación con los atributos de fecha de la tabla Lugares.
- Por otra parte, entendemos que la implementación utilizada para la tabla Lugares no fue la más adecuada. Por los atributos que la componen, la tabla debería llamarse Eventos (o algún otro nombre que haga referencia a sucesos que ocurrieron en un lugar y momento determinado). Además debería existir una tabla Lugares que contenga atributos específicos de lugares (ubicación, capacidad, nivel, etc.).
- Otra recomendación para mejorar la tabla Lugares consiste en agregar atributos añoFin y mesFin dado que sin estos atributos no es posible asignar fecha de fin a aquellos congresos que culminaron en un mes/año distintos a los de su fecha de inicio.

## **Restricciones**

Para cumplir con las restricciones indicadas en la letra se aplicaron instrucciones ALTER sobre las tablas creadas a partir del script de creación analizado en el punto anterior.

También se crearon los siguientes disparadores:

- Disparador para generar el identificador alfanumérico de los trabajos (Tabla Trabajo)
- Disparador para controlar que un INVESTIGADOR no cambie de UNIVERSIDAD (Tabla Investigador)
- Disparador para que un trabajo no se referencie a sí mismo (Tabla Referencias)
- Disparador para controlar que diaFin no sea nulo cuando tipolugar toma el valor 'Congresos' (Tabla Lugares)

## **Índices**

Para la creación de índices se siguió el criterio definido en clase: se crearon índices sobre las claves foráneas en aquellos casos que no ocupaban el primer lugar de la clave primaria.

## **Inserción de datos de prueba**

Para la inserción de datos de prueba se realizaron inserciones múltiples en cada tabla (excepto en las tablas Trabajo y Referencias) prestando especial atención a la calidad de los datos ingresados. Esto quiere decir que los juegos de datos ingresados buscan probar los distintos aspectos de la solución propuesta.

El script con los juegos de datos que, dadas las restricciones aplicadas, serán rechazados (o no serán insertados) se muestran en el anexo Datos a rechazar.

En el caso de la tabla Trabajo se realizaron inserciones individuales para evitar conflictos con el disparador que genera el identificador alfanumérico.

## Procedimientos y funciones

- Para el caso del *Se pide 4-a* se consideró que la fecha de publicación de los trabajos fue la misma que la fecha de inicio. Optamos por esta solución ya que, como lo mencionamos más arriba, la tabla Trabajo no contiene ningún atributo para designar esta fecha.
- En el *Se pide 4-d* implementamos un procedimiento dado que la operación a ejecutar es una actualización.
- En el *Se pide 4-h* implementamos una función ya que la instrucción requerida es de tipo SELECT.

## Disparadores

Dado que el disparador del *Se pide 5-a* es un disparador de tipo `INSTEAD OF` sobre la tabla Trabajo, incluimos en este disparador el disparador que genera el identificador alfanumérico sobre la tabla mencionada.

Para el *Se pide 5-b* se consideró que un trabajo no fue publicado si el atributo `lugarPublic` de la tabla Trabajo es nulo.

En el *Se pide 5-c*, dado que la letra es confusa, implementamos un disparador que sólo permite insertar dos trabajos, uno como referencia de otro, si tienen palabras claves en común. Para eliminar todos los registros vinculados (en todas las tablas) al trabajo a eliminar, primero se eliminan los registros de la tabla Referencias, luego de TAutores y TTags y por último de la tabla Trabajo.

Para el log solicitado en *Se pide 5-d* se creó una tabla auxiliar `LogInsertAndUpdate`.

## Consultas SQL

En la consulta del *Se pide 6-c*, seleccionamos el nombre de la universidad por entender que este nombre identifica a la universidad y cumple con el objetivo de la consulta solicitada.

Para simplificar la consulta del *Se pide 6-e* (y evitar la duplicación de código) creamos una función que, dados un nivel de lugar y un id de investigador de la carrera de ingeniería, retorna la cantidad de trabajos del investigador recibido y publicados en un lugar con el nivel recibido en los últimos 5 años.

## **Vistas**

En la vista creada para el *Se pide 7-b* se utilizaron las funciones de agregación `MIN()` y `MAX()`. Cuando un investigador tiene un sólo trabajo de un mismo tipo, en ese caso las fechas coinciden pues, ese trabajo cumple la condición de ser el primer y último trabajo de ese tipo para ese autor.

Script con las restricciones de integridad creadas sobre el script de creación de tablas, índices, y el ingreso de datos de prueba

```
/*#####*/
/*          ALTERACIONES PARA AGREGAR RESTRICCIONES          */
/*#####*/

/* UNIVERSIDAD */

ALTER TABLE Universidad
ALTER COLUMN nombre VARCHAR(100) NOT NULL;
GO

ALTER TABLE Universidad
ADD Constraint pk_nombre Primary key(nombre)
GO

ALTER TABLE Universidad
ADD telefono varchar(20) not null;
GO

/*-----*/

/* INVESTIGADOR */
ALTER TABLE Investigador
DROP COLUMN idInvestigador
GO

ALTER TABLE Investigador
ADD idInvestigador INT NOT NULL IDENTITY(1,1)
GO

ALTER TABLE Investigador
ADD idUniversidad VARCHAR(100) NOT NULL
GO

ALTER TABLE Investigador
ADD CONSTRAINT Investigador_PK PRIMARY KEY (idInvestigador)
GO
```

```
ALTER TABLE Investigador
ADD CONSTRAINT Investigador_FK FOREIGN KEY (idUniversidad)
REFERENCES Universidad
GO
```

```
ALTER TABLE Investigador
ADD CONSTRAINT NivelInv_CH CHECK (nivelInvestig IN ('EGrado',
'EMaestria', 'EDoctor', 'Doctor'))
GO
```

```
ALTER TABLE Investigador
ADD UNIQUE (Mail)
GO
```

```
ALTER TABLE Investigador
ALTER COLUMN cantTrabPub INT NOT NULL
GO
```

```
/*-----*/
/*Disparador para controlar que un INVESTIGADOR no cambie de
UNIVERSIDAD */
```

```
CREATE TRIGGER INVESTIGADOR_UNIVERSIDAD
ON Investigador
INSTEAD OF UPDATE
AS
BEGIN
    IF(EXISTS
        (
            SELECT *
            FROM Investigador x, inserted i
            WHERE x.idInvestigador = i.idInvestigador
            AND x.idUniversidad <> i.idUniversidad
        )
    )
    BEGIN
        PRINT 'No se admiten cambios de UNIVERSIDAD para un
INVESTIGADOR.'
    END
    ELSE
    BEGIN
```



```

        UPDATE Investigador
        SET nombre = i.nombre, mail = i.mail, telefono =
i.telefono, carrera = i.carrera, nivelInvestig =
i.nivelInvestig,cantTrabPub = i.cantTrabPub,idUniversidad =
i.idUniversidad
        FROM inserted i, Investigador inv
        where i.idInvestigador = inv.idInvestigador
    END
END
GO

/*-----*/

/* TRABAJO */
ALTER TABLE Trabajo
DROP COLUMN idTrab
GO

ALTER TABLE Trabajo
ADD idTrab varchar(10) not null;
GO

ALTER TABLE Trabajo
ALTER COLUMN descripTrab VARCHAR(200)
GO

ALTER TABLE Trabajo
ADD CONSTRAINT tipoTrab_check CHECK (tipoTrab IN ('poster',
'articulo', 'capitulo', 'otro'))
GO

ALTER TABLE Trabajo
ADD CONSTRAINT Trabajo_PK PRIMARY KEY (idTrab)
GO

ALTER TABLE Trabajo
ADD CONSTRAINT Trabajo_FK FOREIGN KEY (lugarPublic)
REFERENCES Lugares
GO

/*-----*/
/*Disparador para generar ID Trabajo */

```

```

CREATE TRIGGER trig_idTrab
ON Trabajo
INSTEAD OF INSERT
AS
BEGIN
    IF( NOT EXISTS( Select COUNT(*) From inserted GROUP BY nomTrab
having count(*) > 1))
        BEGIN
            DECLARE @ultINS int;
            SET @ultINS = (select COUNT(*) from Trabajo where tipoTrab
in (select tipoTrab from inserted));

            DECLARE @alphaNumID varchar(10);
            SELECT @alphaNumID = UPPER(SUBSTRING(tipoTrab, 1, 1)) from
inserted;

            SET @alphaNumID = @alphaNumID + CONVERT(varchar(10),
@ultINS);

            INSERT INTO Trabajo
            SELECT nomTrab, descripTrab, tipoTrab, fechaInicio,
linkTrab, lugarPublic, @alphaNumID
            FROM inserted
        END
    ELSE
        BEGIN
            PRINT 'No se admiten inserciones múltiples.'
        END
END
GO

/*-----*/

/* TAGS */
ALTER TABLE Tags
DROP COLUMN idTag
GO

ALTER TABLE Tags
ADD idTag INT IDENTITY(1,2) NOT NULL
GO

```

```
ALTER TABLE Tags
ADD CONSTRAINT Tags_PK PRIMARY KEY (idTag)
GO
```

```
ALTER TABLE Tags
ALTER COLUMN palabra varchar(50) NOT NULL
GO
```

```
/*-----*/
```

```
/* TTAGS */
ALTER TABLE TTags
ADD CONSTRAINT TTags_PK PRIMARY KEY (idTrab, idTag)
GO
```

```
ALTER TABLE TTags
ADD CONSTRAINT TTags_FK1 FOREIGN KEY (idTrab)
REFERENCES Trabajo
GO
```

```
ALTER TABLE TTags
ADD CONSTRAINT TTags_FK2 FOREIGN KEY (idTag)
REFERENCES Tags
GO
```

```
/*-----*/
```

```
/* TAUtores */
ALTER TABLE TAutores
ADD CONSTRAINT TAutores_PK PRIMARY KEY (idTrab, idInvestigador)
GO
```

```
ALTER TABLE TAutores
ADD CONSTRAINT TAutores_FK1 FOREIGN KEY (idTrab)
REFERENCES Trabajo
GO
```

```
ALTER TABLE TAutores
ADD CONSTRAINT TAutores_FK2 FOREIGN KEY (idInvestigador)
REFERENCES Investigador
GO
```

```
ALTER TABLE TAutores
ALTER COLUMN rolinvestig varchar(20);
GO
```

```
ALTER TABLE TAutores
ADD CONSTRAINT rolinvestig_check CHECK (rolinvestig IN
('autor-ppal', 'autor-sec', 'autor-director'))
GO
```

```
/*-----*/
```

```
/* REFERENCIAS */
ALTER TABLE Referencias
ADD CONSTRAINT Referencias_PK PRIMARY KEY (idTrab,
idTrabReferenciado)
GO
```

```
ALTER TABLE Referencias
ADD CONSTRAINT Referencias_FK_Trab FOREIGN KEY (idTrab)
REFERENCES Trabajo
GO
```

```
ALTER TABLE Referencias
ADD CONSTRAINT Referencias_FK_TrabRef FOREIGN KEY
(idTrabReferenciado)
REFERENCES Trabajo
GO
```

```
ALTER TABLE Referencias
ADD CONSTRAINT Referencias_Referencia CHECK (idTrab <>
idTrabReferenciado)
```

```
/*-----*/
```

```
/* LUGARES */
ALTER TABLE Lugares
ALTER COLUMN nombre varchar(250) not null;
GO
```

```
ALTER TABLE Lugares
ADD CONSTRAINT nombre_uniq unique (nombre);
```

GO

```
ALTER TABLE Lugares
ADD tipoLugar varchar(10) not null;
GO
```

```
ALTER TABLE Lugares
ADD CONSTRAINT tipoLugar_check CHECK (tipoLugar IN ('Congresos',
'Revistas', 'Libros'))
GO
```

```
ALTER TABLE Lugares
ADD CONSTRAINT nivelLugar_check CHECK (nivelLugar BETWEEN 1 and 4);
GO
```

```
ALTER TABLE Lugares
ALTER COLUMN universidad VARCHAR(100) NOT NULL
GO
```

```
ALTER TABLE Lugares
ADD CONSTRAINT Lugares_FK FOREIGN KEY (universidad)
REFERENCES Universidad
GO
```

```
ALTER TABLE Lugares
ADD CONSTRAINT mes_check CHECK (mes BETWEEN 1 and 12)
GO
```

```
ALTER TABLE Lugares
ADD CONSTRAINT diaI_check CHECK (diaIni BETWEEN 1 and 31)
GO
```

```
ALTER TABLE Lugares
ADD CONSTRAINT diaF_check CHECK (diaFin BETWEEN 1 and 31)
GO
```

```
ALTER TABLE Lugares
ADD CONSTRAINT año_check CHECK (año BETWEEN 1900 and YEAR(
GETDATE()));
GO
```

```
/*-----*/  
/* Disparador para controlar que diaFin no sea nulo cuando tipolugar  
tiene valor 'Congresos' en tabla LUGARES*/
```

```
CREATE TRIGGER EvitarDiaFinNulo_LUGARES
```

```
ON Lugares
```

```
INSTEAD OF INSERT
```

```
AS
```

```
BEGIN
```

```
    DECLARE @anio VARCHAR (4),  
            @mes VARCHAR (2),  
            @diaIni VARCHAR (2),  
            @diaFin VARCHAR (10),  
            @fechaInicio DATE,  
            @fechaFin DATE,  
            @fechaActual DATE
```

```
    SELECT @anio = CAST(año AS VARCHAR(4)) FROM inserted
```

```
    SELECT @mes = CAST(mes AS VARCHAR(2)) FROM inserted
```

```
    SELECT @diaIni = CAST(diaIni AS varchar(2)) FROM inserted
```

```
    SET @fechaInicio = CONVERT(date,  
@anio+'-'+@mes+'-'+@diaIni)
```

```
    IF(EXISTS (SELECT tipoLugar from inserted WHERE tipoLugar  
NOT LIKE 'Congresos') AND EXISTS(SELECT diaFin FROM inserted WHERE  
diafin IS NULL AND tipoLugar NOT LIKE 'Congresos'))
```

```
    BEGIN
```

```
        INSERT Lugares
```

```
        SELECT idLugar = inserted.idLugar, nombre =  
inserted.nombre, nivellugar = inserted.nivellugar, año =  
inserted.año, mes = inserted.mes, diaIni = inserted.diaIni, diaFin =  
null, link = inserted.link, universidad = inserted.universidad,  
tipolugar = inserted.tipolugar
```

```
        FROM inserted
```

```
    END
```

```
    ELSE
```

```
    IF(EXISTS (SELECT tipoLugar from inserted WHERE tipoLugar  
LIKE 'Congresos') AND NOT EXISTS (SELECT diaFin FROM inserted WHERE  
diafin IS NULL AND tipoLugar LIKE 'Congresos'))
```

```
    BEGIN
```

```

                SELECT @diaFin = CAST(diaFin AS VARCHAR(2)) FROM
inserted
                SET @fechaFin = CONVERT(date,
@anio+'-'+@mes+'-'+@diaFin)
                SET @fechaActual = GETDATE()
                IF(@fechaInicio < @fechaFin AND @fechaInicio <
@fechaActual)
                BEGIN
                        INSERT Lugares
                        SELECT idLugar = inserted.idLugar, nombre =
inserted.nombre, nivelLugar = inserted.nivelLugar, año =
inserted.año, mes = inserted.mes, diaIni = inserted.diaIni, diaFin =
inserted.diaFin, link = inserted.link, universidad =
inserted.universidad, tipoLugar = inserted.tipoLugar
                        FROM inserted
                END
                ELSE
                BEGIN
                        PRINT 'La fecha de inicio debe ser anterior a
la fecha fin y ambas deben ser anteriores a la fecha actual.'
                END
                END
                ELSE
                BEGIN
                        PRINT 'Todos los congresos deben tener un día fin.'
                END
                END
END
GO

```

```

/*#####*/
/*#####*/
/*#####*/
/*
                                SE PIDE #2                                */
/*#####*/
/*#####*/
/*#####*/
/* 2. Creación de índices que considere puedan ser útiles para
optimizar las consultas (según criterio establecido en el curso)*/

```

```

CREATE INDEX i_investigador_uni ON Investigador(idUniversidad);
CREATE INDEX i_lugares_uni ON Lugares(universidad);
CREATE INDEX i_autores_investigador ON TAutores(idInvestigador);
CREATE INDEX i_trabajo_lugar ON Trabajo(lugarPublic);
CREATE INDEX i_trabajo_referenciado ON
Referencias(idTrabReferenciado);
CREATE INDEX i_ttags_tags ON TTags(idTag);
GO

```

```

/*#####*/
/*#####*/
/*#####*/
/*
                                SE PIDE #3                                */
/*#####*/
/*#####*/
/*#####*/
/* 3. Ingreso de un juego completo de datos de prueba (será más
valorada la calidad de los datos más que la cantidad. El mismo
debería incluir ejemplos que deban ser rechazados por no
cumplir con las restricciones implementadas.*/

```

```

/*+++++*/
/*
                                Tabla UNIVERSIDAD                                */
/*+++++*/

```

```

/*Datos OK*/
INSERT INTO Universidad
VALUES
('Udelar', 'Uruguay', 'Montevideo', '29009999'),
('ORT', 'Uruguay', 'Montevideo', '27007777'),
('UCUDAL', 'Uruguay', 'Montevideo', '24004444'),
('UM', 'Uruguay', 'Montevideo', '26006666'),

```



```
( 'Universidad de Palermo', 'Argentina', 'Buenos Aires', '44116666'),
( 'UBA', 'Argentina', 'Buenos Aires', '44110000'),
( 'Universidad de Córdoba', 'Argentina', 'Córdoba', '45000000'),
( 'Universidad de Brasilia', 'Brasil', 'Brasilia', '62501253'),
( 'Universidad Federal de Alagoas', 'Brasil', 'Alagoas', '78514569'),
( 'Universidad de Amazonas', 'Brasil', 'Amazonas', '35358978')
```

```
/*+++++*/
/*                               Tabla LUGARES                               */
/*+++++*/
```

```
/* Datos OK */
```

```
INSERT INTO Lugares
VALUES
```

```
(1, 'Teatro Solis', 4, 2016, 11, 8, null,
'http://www.teatrosolis.org.uy', 'Udelar', 'Revistas'),
(2, 'LATU', 4, 2015, 11, 9, 13, 'www.latu.org.uy', 'ORT',
'Congresos'),
(3, 'Radisson Victoria Plaza Hotel', 4, 2015, 5, 20, null,
'https://www.radissonblu.com', 'UM', 'Libros'),
(4, 'Holiday Inn', 2, 2017, 2, 10, 16, 'https://www.ihg.com',
'UCUDAL', 'Congresos'),
(5, 'Hotel Dazzler', 1, 2017, 8, 8, null,
'https://www.dazzlerhoteles.com', 'UBA', 'Revistas')
```

```
/*+++++*/
/*                               Tabla INVESTIGADOR                               */
/*+++++*/
```

```
/*Datos OK*/
```

```
INSERT INTO Investigador (nombre, mail, telefono, carrera,
nivelInvestig,cantTrabPub,idUniversidad)
```

```
VALUES
```

```
( 'Marcelo López', 'mlopez@investigadores.com.uy', '098999999',
'Ingeniería Química', 'EGrado', 5,'Udelar'),
( 'Laura Marquisio', 'lmarqui@investigadores.com.uy', '098111111',
'Licenciatura en Relaciones Internacionales', 'EMaestria', 1,'UM'),
( 'Fabián Méndez', 'fmendez@investigadores.com.uy', '095951135',
'Licenciatura en Economía', 'EMaestria', 3,'Udelar'),
```

```
( 'Silvia Luque', 'sluque@investigadores.com.uy', '096457215',
'Licenciatura en Economía', 'EDoctor', 9, 'Udelar'),
( 'Silvio Duarte', 'sduarte@investigadores.com.uy', '099485245',
'Licenciatura en Sistemas', 'Doctor', 15, 'Udelar'),
( 'Walter Clitish', 'wclitish@investigadores.com.uy', '099123456',
'Licenciatura en Letras', 'EDoctor', 12, 'Udelar'),
( 'Maicol Uriarte', 'muriarte@investigadores.com.uy', '15648524',
'Licenciatura en Bellas Artes', 'EMaestria', 9, 'UBA'),
( 'Marianela Ifrán', 'mifran@investigadores.com.uy', '15677425',
'Ingeniería Naval', 'EGrado', 1, 'Universidad de Córdoba'),
( 'Linda Cibils', 'lcibils@investigadores.com.uy', '48579568',
'Licenciatura en Ciencias Biológicas', 'EDoctor', 5, 'Universidad de
Amazonas'),
( 'Marcio Avellanal', 'mavellanal@investigadores.com.uy', '45129685',
'Licenciatura en Matemáticas', 'EDoctor', 5, 'Universidad Federal de
Alagoas'),
( 'Guillermo Polachek', 'polachek@ort.edu.uy', '0911111111',
'Ingeniería', 'Doctor', 50, 'ORT'),
( 'Sebastian Villar', 'villar@uamazonas.edu.uy', '0922222222',
'Ingeniería', 'Doctor', 50, 'Universidad de Amazonas'),
( 'Atuagualpo Galpones', 'atagua@ugmail.com', '0933333333',
'Ingeniería', 'Doctor', 3, 'ORT')
```

```
/*+++++*/
/*                               Tabla TRABAJO                               */
/*+++++*/
```

```
/* Datos OK */
```

```
INSERT INTO Trabajo
VALUES ('Investigacion GARZA CUCA', 'La garza cuca o también
denominada garza mora (Ardea cocoi) es un ave nativa del Centro y
Sudamérica, se estudia su ambiente y entorno', 'articulo',
'2016-04-03',
'https://www.infoanimales.com/informacion-sobre-la-garza-cuca',1,
'id')
```

```
INSERT INTO Trabajo
VALUES ('Venado de campo', 'Investigacion sobre el venado de campo,
uno de los integrantes más característicos de la fauna uruguaya',
'capitulo', '2017-02-01',
'http://blogs.ceibal.edu.uy/formacion/coleccion-de-recursos/venado
-de-campo/',2, 'id')
```

```
INSERT INTO Trabajo
VALUES ('Investigacion sobre el Agua', 'El agua es un bien y un
recurso cada vez mas escaso, que debe ser valorado, protegido y
recuperado', 'poster', '2017-05-17',
'https://es.slideshare.net/sssanchezayelen/investigacin-sobre-el-agu
a',3, 'id')
```

```
INSERT INTO Trabajo
VALUES ('Investigacion sobre medio ambiente ', 'El análisis de lo
ambiental desde la perspectiva de lo social', 'Otro', '2017-08-20',
'http://cis.ufro.cl/index.php?option=com_content&view=article&id=45&
Itemid=34',5, 'id')
```

```
INSERT INTO Trabajo
VALUES ('Investigacion sobre las drogas', 'La drogadicción es una
enfermedad que consiste en la dependencia de sustancias que afectan
el sistema nervioso central y las funciones cerebrales', 'articulo',
'2017-05-17',
'https://www.monografias.com/docs/Investigacion-sobre-las-drogas-FKJ
QBHKYMZ',4, 'id')
```

```
INSERT INTO Trabajo
VALUES ('Investigacion sobre Cultura maya ', 'La civilización maya
es sin duda la más fascinante de las antiguas culturas americanas',
'Otro', '2017-04-28',
'https://www.biografiasyvidas.com/historia/cultura_maya.htm',5,
'id')
```

```
INSERT INTO Trabajo
VALUES ('Investigacion sobre SQL', 'Lenguaje específico del dominio
que da acceso a un sistema de gestión de bases de datos
relacionales', 'articulo', '2017-12-08',
'https://es.wikipedia.org/wiki/SQL',5, 'id')
```

```
INSERT INTO Trabajo
VALUES ('Investigacion sobre XQuery', 'XQuery es un lenguaje de
consulta para fuentes de datos XML', 'articulo', '2017-12-07',
'https://es.wikipedia.org/wiki/XQuery',5, 'id')
```

```
INSERT INTO Trabajo
VALUES ('Investigacion sobre Bases de Datos Oracle', 'Oracle
Database es un sistema de gestión de base de datos de tipo
```

```
objeto-relacional', 'articulo', '2017-12-06',  
'https://es.wikipedia.org/wiki/Oracle_Database',5, 'id')
```

```
INSERT INTO Trabajo  
VALUES ('Investigacion sobre Bases de Datos', 'Una base de datos o  
banco de datos es un conjunto de datos pertenecientes a un mismo  
contexto y almacenados sistemáticamente para su posterior uso',  
'capitulo', '2017-12-08',  
'https://es.wikipedia.org/wiki/Base_de_datos',5, 'id')
```

```
INSERT INTO Trabajo  
VALUES('Investigacion sobre Windows', 'Microsoft Windows es un  
sistema operativo, es decir, un conjunto de programas que posibilita  
la administración de los recursos de una computadora', 'articulo',  
'2017-11-10', 'https://definicion.de/windows/',4, 'id')
```

```
INSERT INTO Trabajo  
VALUES ('Investigacion sobre SO. Ubuntu', 'Ubuntu es una  
distribución del sistema operativo GNU/Linux y que se distribuye  
como software libre', 'articulo', '2017-11-09',  
'https://es.wikipedia.org/wiki/Ubuntu',4, 'id')
```

```
INSERT INTO Trabajo  
VALUES ('Investigacion sobre Sistemas Operativos', 'Un sistema  
operativo es el software principal o conjunto de programas de un  
sistema informático que gestiona los recursos de hardware y provee  
servicios a los programas de aplicación de software', 'articulo',  
'2017-11-08', 'https://es.wikipedia.org/wiki/Sistema_operativo',4,  
'id')
```

```

/*+++++*/
/*                               Tabla TAGS                               */
/*+++++*/

```

```

/*Datos OK*/
INSERT INTO Tags
VALUES
('garza'),
('cuca'),
('venado'),
('campo'),
('fauna'),
('animales'),
('silvestre'),
('agua'),
('ambiente'),
('ecología'),
('drogas'),
('adicciones'),
('cultura'),
('mayas'),
('BASE DE DATOS'),
('sql'),
('xquery'),
('oracle'),
('windows'),
('ubuntu'),
('sistema-operativo')

```

```

/*+++++*/
/*                               Tabla TTAGS                               */
/*+++++*/

```

```

/*Datos OK*/
INSERT INTO TTags
VALUES
('A0',1),
('A0',3),
('C0',5),
('00',7),
('01',9),
('P0',15),

```

```

('01',27),
('A2',29),
('A3',29),
('A4',29),
('C1',29),
('A2',31),
('A3',33),
('A4',35),
('A5',41),
('A6',41),
('A7',41),
('A5',37),
('A6',39)

```

```

/*+++++*/
/*                               Tabla TAUtores                               */
/*+++++*/

```

```

INSERT INTO TAutores
VALUES
('A0',1,'autor-ppal'),
('A0',5,'autor-sec'),
('A6',13,'autor-sec'),
('C1',13,'autor-sec'),
('C0',2,'autor-director'),
('01',2,'autor-ppal'),
('P0',3,'autor-director'),
('A1',4,'autor-ppal'),
('A1',6,'autor-ppal'),
('A1',9,'autor-sec'),
('A0',9,'autor-sec'),
('A1',8,'autor-director'),
('A2',11,'autor-ppal'),
('A2',12,'autor-ppal'),
('A3',11,'autor-ppal'),
('A3',12,'autor-ppal'),
('A4',11,'autor-ppal'),
('A4',12,'autor-ppal'),
('C1',11,'autor-ppal'),
('C1',12,'autor-ppal'),
('A5',11,'autor-ppal'),

```

```
('A5',12,'autor-ppal'),
('A6',11,'autor-ppal'),
('A6',12,'autor-ppal'),
('A7',11,'autor-ppal'),
('A7',12,'autor-ppal')
```

```
/*+++++*/
/*                               Tabla REFERENCIAS                               */
/*+++++*/
/* Datos OK */
INSERT INTO Referencias
VALUES('A0','C0')
INSERT INTO Referencias
VALUES('P0','00')
INSERT INTO Referencias
VALUES('A0','A1')
INSERT INTO Referencias
VALUES('C1','A2')
INSERT INTO Referencias
VALUES('C1','A3')
INSERT INTO Referencias
VALUES('C1','A4')
INSERT INTO Referencias
VALUES('A7','A6')
INSERT INTO Referencias
VALUES('A7','A5')
```

GO

## Script con la creación de disparadores

```
/*#####*/
/*#####*/
/*#####*/
/*          SE PIDE #5          */
/*#####*/
/*#####*/
/*#####*/

/*
5 - a. Crear un trigger que al insertar un trabajo asegure las
restricciones identificadas, y en caso de que no se asignó una fecha
de inicio el trigger asigne el primero de enero del año actual
como fecha de inicio del trabajo.
En el caso de considerarse insert de a un trabajo por vez, debe
asegurarse de que la operación se haga solo en ese caso.*/

DROP TRIGGER trig_idTrab
go

CREATE TRIGGER tg_RestriccionesTrabajo
ON Trabajo
INSTEAD OF INSERT
AS
BEGIN
    IF( NOT EXISTS( Select COUNT(*) From inserted GROUP BY nomTrab
having count(*) > 1))
        BEGIN
            DECLARE @nomTrab VARCHAR(100);
            DECLARE @descripTrab VARCHAR(100);
            DECLARE @tipoTrab VARCHAR(20);
            DECLARE @fecha date;
            /*Trigger Trabajo alphanumerico*/
            declare @ultINS int;
            set @ultINS = (select COUNT(*) from Trabajo where
tipoTrab in (select tipoTrab from inserted));
            declare @alphaNumID varchar(10);
            select @alphaNumID = UPPER(SUBSTRING(tipoTrab, 1, 1))
from inserted;
            set @alphaNumID = @alphaNumID + CONVERT(varchar(10),
@ultINS);
```



```

/* Trigger 5a */
select @nomTrab = nomTrab, @descripTrab = descripTrab, @tipoTrab =
tipoTrab, @fecha = fechaInicio
        from inserted

        IF(@nomTrab is not null AND @tipoTrab is not null)
        BEGIN
            IF(@tipoTrab like 'poster' OR @tipoTrab like
'articulo' OR @tipoTrab like 'capitulo' OR @tipoTrab like 'otro')
            BEGIN
                if(@fecha is null) Begin set @fecha =
'01-01-' + CONVERT(varchar(4), YEAR(GETDATE())); End
                insert into Trabajo
                select nomTrab, descripTrab, tipoTrab, @fecha,
linkTrab, lugarPublic, @alphaNumID
                from inserted
            END
            ELSE
            BEGIN
                PRINT 'Parametros no correctos'
            END
        END

    END
    ELSE
    BEGIN
        PRINT 'No se admiten inserciones múltiples.'
    END
END
GO

/*
5b - Crear un trigger que controle que solo puedan eliminarse
trabajos no publicados que iniciaron hace más de 2 años. Eliminando
todos los datos de la base de datos que considere necesarios
asociados a dichos trabajos. Debe considerarse eliminaciones
múltiples.
*/

CREATE TRIGGER tg_EliminarTrabajos

```

```

ON Trabajo
INSTEAD OF DELETE
AS
BEGIN

    DELETE FROM Referencias
    WHERE idTrab IN (
        SELECT x.idTrab
        FROM Trabajo x
        WHERE YEAR(x.fechaInicio) < (YEAR(GETDATE()) - 2)
    )
    AND idTrab IN (
        SELECT y.idTrab
        FROM Trabajo y
        WHERE y.lugarPublic IS NULL
    )
    AND idTrab IN(
        SELECT idTrab
        FROM deleted
    )

    DELETE FROM Referencias
    WHERE idTrabReferenciado IN(
        SELECT x.idTrab
        FROM Trabajo x
        WHERE YEAR(x.fechaInicio) < (YEAR(GETDATE()) - 2)
    )
    AND idTrabReferenciado IN (
        SELECT y.idTrab
        FROM Trabajo y
        WHERE y.lugarPublic IS NULL
    )
    AND idTrabReferenciado IN(
        SELECT idTrab
        FROM deleted
    )

    DELETE FROM TAutores
    WHERE idTrab IN (
        SELECT x.idTrab
        FROM Trabajo x
        WHERE YEAR(x.fechaInicio) < (YEAR(GETDATE()) - 2)
    )

```

```

)
AND idTrab IN (
    SELECT y.idTrab
    FROM Trabajo y
    WHERE y.lugarPublic IS NULL
)
AND idTrab IN(
    SELECT idTrab
    FROM deleted
)

DELETE FROM TTags
WHERE idTrab IN (
    SELECT x.idTrab
    FROM Trabajo x
    WHERE YEAR(x.fechaInicio) < (YEAR(GETDATE()) - 2)
)
AND idTrab IN (
    SELECT y.idTrab
    FROM Trabajo y
    WHERE y.lugarPublic IS NULL
)
AND idTrab IN(
    SELECT idTrab
    FROM deleted
)

DELETE FROM Trabajo
WHERE YEAR(fechaInicio) < (YEAR(GETDATE()) - 2)
AND lugarPublic IS NULL
AND idTrab IN(
    SELECT idTrab
    FROM deleted
)
END
GO

```

/\* 5c - Crear un trigger que solo permita insertar dos trabajos, uno como referencia del otro, si tienen algún tema (palabras claves) en común. \*/

```
CREATE TRIGGER tg_InsertarReferencias
ON Referencias
INSTEAD OF INSERT
AS
BEGIN

    DECLARE @trabajo VARCHAR(10)

    SELECT @trabajo = x.idTrab
    FROM inserted x
    WHERE idTrab IN(
        SELECT idTrab
        FROM TTags
        WHERE idTag IN
        (
            SELECT idTag
            FROM TTags
            WHERE idTrab IN(
                SELECT y.idTrabReferenciado
                FROM inserted y
            )
        )
    )

    IF (@trabajo IS NOT NULL)
    BEGIN
        INSERT INTO Referencias
        SELECT idtrab, idTrabReferenciado
        FROM inserted
    END
    ELSE
    BEGIN
        PRINT 'Los trabajos no tienen temas (palabras clave) en
común.'
    END
END
GO
```

```

/* 5d - Crear un trigger que audite las inserciones y las
actualizaciones de los lugares que tienen algún trabajo publicado,
registrando en alguna tabla auxiliar el lugar que se quiere
actualizar/insertar, la operación (INS o UPD), el usuario y la fecha
hora de la operación.*/

```

```

CREATE TABLE LogInsertAndUpdate (
    lugar INT,
    operacion VARCHAR(3) CHECK (operacion IN ('INS', 'UPD')),
    usuario VARCHAR(50),
    fechaOperacion date
)
GO

```

```

CREATE TRIGGER tg_AuditarInsOrUpd
ON Lugares
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @lugar INT
    SELECT @lugar = idLugar
    FROM inserted

    IF(EXISTS (SELECT * FROM inserted) AND NOT EXISTS (SELECT
* FROM deleted))
    BEGIN
        INSERT INTO LogInsertAndUpdate
        VALUES (@lugar, 'INS', USER, GETDATE())
    END
    ELSE IF(EXISTS (SELECT * FROM inserted) AND EXISTS
(SELECT * FROM deleted))
    BEGIN

        IF(EXISTS (
        SELECT *
        FROM inserted
        WHERE idLugar IN
        (
            SELECT lugarPublic
            FROM Trabajo
        )
        )
    )

```

```
)  
BEGIN  
        INSERT INTO LogInsertAndUpdate  
VALUES (@lugar, 'UPD', USER, GETDATE())  
END  
END  
GO
```

## Script con la creación de funciones y procedimientos

```
/*#####*/
/*#####*/
/*#####*/
/*
                                SE PIDE #4
                                */
/*#####*/
/*#####*/
/*#####*/
```

/\* 4a - Crear una función que dada una universidad devuelva el último trabajo publicado por esta. Si hay más de uno devolver uno cualquiera.\*/

```
CREATE FUNCTION fn_UltimoTrabajoPorUniv (@unaUniversidad
VARCHAR(100))
RETURNS VARCHAR(110)
AS
BEGIN
    DECLARE @ultTrabajo VARCHAR(10)
    DECLARE @nomTrabajo VARCHAR(100)
    DECLARE @ret VARCHAR(110)

    SELECT @ultTrabajo = idTrab , @nomTrabajo = nomTrab
    FROM Trabajo tr
    WHERE lugarPublic IN
    (
        SELECT idLugar
        FROM Lugares
        WHERE universidad = @unaUniversidad
    )AND tr.fechaInicio in (Select Max(fechaInicio) from Trabajo
WHERE lugarPublic IN
    (
        SELECT idLugar
        FROM Lugares
        WHERE universidad = @unaUniversidad
    ))

    set @ret = @ultTrabajo + ' - ' + @nomTrabajo;

RETURN @ret
```

END

GO

```
declare @ultTrabUni VARCHAR(110);
set @ultTrabUni = dbo.fn_UltimoTrabajoPorUniv('UBA')
PRINT @ultTrabUni
GO
```

/\* 4b - Crear una función almacenada que reciba como parámetro un trabajo y devuelva la cantidad de referencias externas que tiene.\*/

```
CREATE FUNCTION fn_CantReferenciasExt ( @unTrabajo VARCHAR(100))
RETURNS INT
AS
BEGIN
    DECLARE @cantReferencias INT
    IF(EXISTS (      SELECT * FROM Referencias r WHERE r.idTrab =
@unTrabajo))
        BEGIN
            SELECT @cantReferencias = COUNT(*)
            FROM Referencias r
            WHERE r.idTrab = @unTrabajo
            AND r.idTrabReferenciado NOT IN(
                SELECT idTrab
                from TAutores a
                where idInvestigador in (
                    select idInvestigador
                    from TAutores b
                    where r.idTrab = b.idTrab
                )
            )
        END
    ELSE
        BEGIN
            SET @cantReferencias = 0
        END

    RETURN @cantReferencias
END
GO
```

```
declare @cantRef int;
```



```

set @cantRef = dbo.fn_CantReferenciasExt('A0')
PRINT @cantRef
go

/*+++++*/

/* 4c - Crear una función que reciba dos investigadores y devuelva
la cantidad de trabajos publicados en los cuales ambos
investigadores fueron autores y alguno de los dos o los dos fueron
autores principales.*/

CREATE FUNCTION fn_CantTrabPublicados
(
    @investigadorA INT,
    @investigadorB INT
)
RETURNS INT
AS
BEGIN
    DECLARE @cantTrabajos INT

    SELECT @cantTrabajos = COUNT (DISTINCT x.idTrab)
    from TAutores x, TAutores y
    where x.idInvestigador = @investigadorA
    and y.idInvestigador = @investigadorB
    and x.idTrab = y.idTrab
    and (x.rolinvestig = 'autor-ppal' or y.rolinvestig=
'autor-ppal')

    RETURN @cantTrabajos
END
GO

declare @cantTrabInvs int;
set @cantTrabInvs = dbo.fn_CantTrabPublicados(11,12)
PRINT @cantTrabInvs

GO
/*+++++*/

```

```

/* 4d - Crear una función o procedimiento, según
considere/corresponda, que dado un investigador actualice el campo
cantidad de trabajos publicados registrados de la tabla
Investigador, y devuelva una indicación de si la cantidad que estaba
registrada (antes de la actualización) era correcta o no.*/

```

```

CREATE PROCEDURE spu_UpdateCantTrab
@unInvestigador INT,
@mensaje VARCHAR(200) output
AS
BEGIN
    DECLARE @cantTrabajosEfectivos INT,
            @cantTrabajosContados INT

    SELECT @cantTrabajosEfectivos = COUNT(*)
    FROM TAutores
    WHERE idInvestigador = @unInvestigador

    SELECT @cantTrabajosContados = cantTrabPub
    FROM Investigador
    WHERE idInvestigador = @unInvestigador

    IF(@cantTrabajosContados = @cantTrabajosEfectivos)
    BEGIN
        SET @mensaje = 'La cantidad de trabajos era correcta, no
fue necesario actualizar la tabla.'
    END
    ELSE
    BEGIN
        UPDATE Investigador
        SET cantTrabPub = @cantTrabajosEfectivos
        WHERE idInvestigador = @unInvestigador
        SET @mensaje = 'La cantidad de trabajos no era correcta,
la tabla se actualizó con el valor correcto.'
    END
END
go

declare @mensaje varchar(200)
EXEC spu_UpdateCantTrab 9, @mensaje output
print @mensaje

```

```

go
/*+++++++*/
/*
4.e Crear una función que dado dos años (año1 y año2) y una palabra
clave devuelva la cantidad de trabajos publicados en libros o
revistas en el rango de años y que trata el tema indicado por la
palabra clave.*/

CREATE FUNCTION fn_CantTrbjAniosClave
(
    @anio1 INT,
    @anio2 INT,
    @clave varchar(50)
)
RETURNS INT
AS
BEGIN
    Declare @cantidad int;
    select @cantidad = COUNT(*)
FROM Trabajo
WHERE idTrab in (
    SELECT idTrab
    From TTags
    where idTag in(
        Select idTag
        from Tags
        where palabra = @clave
    )
) AND lugarPublic in (
    SELECT idLugar
    from Lugares
    where (tipoLugar = 'Revistas' or
tipoLugar = 'Libros')
    AND año BETWEEN @anio1 AND @anio2
)
RETURN @cantidad
END
GO

declare @cantTrabInvs int;

```

```

set @cantTrabInvs = dbo.fn_CantTrbjAniosClave(2015, 2016, 'garza')
PRINT @cantTrabInvs

GO

/*+++++*/
/*
4.f Crear una función que reciba un tipo de trabajo y devuelva un
nuevo identificador para dicho trabajo. */

CREATE FUNCTION fn_NuevoIdentTipoTrabajo
(
    @elTipoTrabajo varchar(50)
)
RETURNS VARCHAR(150)
AS
BEGIN
    DECLARE @ret VARCHAR(150);
    declare @alphaNumID varchar(10);
    IF(@elTipoTrabajo LIKE 'poster' OR @elTipoTrabajo LIKE
'articulo' OR @elTipoTrabajo LIKE 'capitulo' OR @elTipoTrabajo LIKE
'OTRO')
        BEGIN
            select @alphaNumID = UPPER(SUBSTRING(@elTipoTrabajo, 1,
1));

            declare @ultINS int;
            set @ultINS = (select COUNT(*) from Trabajo where
tipoTrab = @elTipoTrabajo);

            set @ret = @alphaNumID + CONVERT(varchar(10), @ultINS);
        END
    ELSE
        BEGIN
            set @ret = 'El parametro ingresado no corresponde a un
tipo de trabajo valido.';
        END
    RETURN @ret
END
GO

declare @nueviTipo varchar(150);
set @nueviTipo = dbo.fn_NuevoIdentTipoTrabajo('articulo')
PRINT @nueviTipo

```

GO

```
/*+++++*/  
/*
```

4.g Crear un procedimiento que dado una universidad devuelva los datos del congreso en el cual público su ultimo Artículo, y algún tipo de indicación de si existe o no un artículo con esas condiciones. Suponemos que no hay congresos que traten los mismos temas y que se solapen en el tiempo. \*/

```
CREATE FUNCTION fn_CongresoDeUniArt
```

```
(
```

```
    @laUniversidad varchar(100)
```

```
)
```

```
RETURNS VARCHAR(300)
```

```
AS
```

```
BEGIN
```

```
    declare @nombre VARCHAR(250);
```

```
    declare @nivelLugar int;
```

```
    declare @diaIni int;
```

```
    declare @mes int;
```

```
    declare @anio int;
```

```
    DECLARE @ret VARCHAR(300);
```

```
        SELECT @nombre = l.nombre, @nivelLugar = l.nivelLugar, @diaIni  
= l.diaIni, @mes = l.mes, @anio = l.año
```

```
        FROM Lugares l, Trabajo t
```

```
        WHERE l.idLugar = t.lugarPublic
```

```
        AND t.idTrab IN (
```

```
            SELECT MAX(idTrab)
```

```
            FROM Trabajo ta
```

```
            WHERE ta.tipoTrab = 'Articulo'
```

```
            AND lugarPublic IN(
```

```
                SELECT idLugar
```

```
                FROM Lugares lg
```

```
                WHERE lg.tipoLugar = 'Congresos'
```

```
                AND lg.universidad = @laUniversidad
```

```
            )
```

```
        )
```

```
    set @ret = @nombre+ ' - Nivel Lugar:'+CONVERT(varchar(1),
```

```
@nivelLugar)+ ' - Fecha: '+CONVERT(varchar(2),
```

```

@diaIni)+'/'+CONVERT(varchar(2), @mes)+'/'+CONVERT(varchar(4),
@anio)
RETURN @ret
END
GO

declare @uni varchar(300);
set @uni = dbo.fn_CongresoDeUniArt('UCUDAL')
PRINT @uni

go
/*+++++*/
/*
4.h Crear un procedimiento o función, según considere/corresponda,
que dado un tipo de trabajo (Articulo, Poster, etc.)
devuelva la cantidad de investigadores que tengan más de 5 trabajos
del tipo indicado, y que tengan la maxima cantidad de publicaciones.
*/

CREATE FUNCTION fn_CantAutTipoTrab
(
    @tipoTrabajo varchar(20)
)
RETURNS int
AS
BEGIN
    declare @ret int;
    SELECT @ret = COUNT(DISTINCT idInvestigador)
    FROM TAutores ta
    WHERE ta.idInvestigador IN(
        SELECT inv.idInvestigador
        FROM Investigador inv
        WHERE inv.cantTrabPub IN(
            SELECT MAX(cantTrabPub)
            FROM Investigador
        )
    )
    AND ta.idInvestigador IN(
        SELECT inv.idInvestigador
        FROM Investigador inv, Trabajo t, TAutores ta
        WHERE inv.idInvestigador = ta.idInvestigador
        AND ta.idTrab = t.idTrab
    )

```

```
        AND t.tipoTrab LIKE @tipoTrabajo
        GROUP BY inv.idInvestigador
        HAVING COUNT(*)>5
    )

    RETURN @ret;
END
go

declare @ret int;
set @ret = dbo.fn_CantAutTipoTrab('articulo')
PRINT @ret
```

## Script con la resolución de las consultas y vista

```
/*#####*/
/*#####*/
/*#####*/
/*
                                SE PIDE #6
                                */
/*#####*/
/*#####*/
/*#####*/
```

```
/* a. Mostrar, para cada trabajo publicado con mas de 3 autores,
el identificador del trabajo, nombre del mismo, la cantidad de
autores que tiene, y lugar donde se publico (id y nombre). */
```

```
SELECT t.idTrab as 'Identificador del trabajo', t.nomTrab as 'Nombre
de Trabajo', Count(distinct ta.idInvestigador) as 'Cantidad de
Autores', l.idLugar as 'Id Lugar', l.nombre as 'Nombe del Lugar'
FROM Trabajo t, Lugares l, TAutores ta
WHERE t.lugarPublic = l.idLugar and t.idTrab = ta.idTrab
GROUP BY t.idTrab,t.nomTrab, l.idLugar, l.nombre
HAVING COUNT(*) > 3
```

```
/* b. Obtener la lista de palabras claves que aparecen en trabajos
sobre 'BASE DE DATOS' (palabra clave) publicados en revistas del año
actual. La salida debe aparecer ordenado alfabéticamente, sin
repetidos. */
```

```
SELECT DISTINCT t.palabra
FROM TTags tt, Tags t, Trabajo w, Lugares l
WHERE tt.idTag = t.idtag
AND tt.idTrab = w.idTrab
AND w.lugarPublic = l.idLugar
AND l.tipoLugar LIKE 'Revistas'
AND l.año = YEAR(GETDATE())
AND w.idTrab IN(
    SELECT z.idTrab
    FROM TTags z
    WHERE z.idTag IN (
        SELECT y.idtag
        FROM Tags y
        WHERE y.palabra = 'BASE DE DATOS'
```



```

    )
)
ORDER BY (t.palabra)
GO

```

```

/* c - Mostrar los datos de las universidades y el link a los
congresos, de las universidades que han realizado más de 2 congresos
de nivel 4, en los ultimos 5 años, con mas de 20 trabajos publicados
en esos congresos. */

```

```

SELECT DISTINCT u.nombre, l.link
FROM Universidad u, Lugares l
WHERE u.nombre = l.universidad
AND l.universidad IN (
    SELECT x.universidad
    FROM Lugares x
    WHERE x.tipoLugar LIKE 'Congresos'
    GROUP BY x.universidad
    HAVING COUNT (*) > 20
)
AND l.nivelLugar = 4
AND l.año > YEAR(GETDATE())-5
AND l.tipoLugar LIKE 'Congresos'
GROUP BY u.nombre, l.link
HAVING COUNT(*) > 2
GO

```

```

/* d- Obtener para cada investigador el ultimo trabajo que inicio
en el cual fue/es autor principal.*/

```

```

SELECT i.idInvestigador, ta.idTrab
FROM Investigador i LEFT OUTER JOIN TAutores ta
ON i.idInvestigador = ta.idInvestigador
LEFT OUTER JOIN Trabajo t
ON ta.idTrab = t.idTrab
WHERE ta.rolinvestig = 'autor-ppal'
AND t.idTrab IN
(
    SELECT TOP 1 x.idTrab
    FROM Trabajo x, TAutores y
    WHERE x.idTrab = y.idTrab
    AND i.idInvestigador = y.idInvestigador
)

```

```
ORDER BY fechaInicio DESC
)
GO
```

/\* e - Para cada investigador mostrar, su identificación, nombre, nombre de la universidad a la que pertenece, y la cantidad de trabajos suyos publicados en lugares de nivel 1, de nivel 2, de nivel 3 y de nivel 4, en los últimos 5 años, en la carrera de Ingeniería. \*/

```
CREATE FUNCTION fn_CantTrabajoPorNivel
(
  @nivelLugar int,
  @idInv int
)
RETURNS int
AS
BEGIN
  DECLARE @ret int
  SELECT @ret = COUNT(*)
  FROM Lugares lu, Investigador inv, TAutores x, Trabajo t
  WHERE inv.idInvestigador = x.idInvestigador
  AND lu.nivelLugar = @nivelLugar
  AND x.idTrab = t.idTrab
  AND inv.idInvestigador = @idInv
  AND t.lugarPublic = lu.idLugar
  AND YEAR(t.fechaInicio) > YEAR(GETDATE())-5
  AND inv.carrera LIKE 'Ingeniería'
  RETURN @ret
END
GO
```

```
SELECT DISTINCT i.idInvestigador, i.nombre, i.idUniversidad,
dbo.fn_CantTrabajoPorNivel(1, i.idInvestigador) as 'Cantidad
trabajos nivel 1',
dbo.fn_CantTrabajoPorNivel(2, i.idInvestigador) as 'Cantidad
trabajos nivel 2',
dbo.fn_CantTrabajoPorNivel(3, i.idInvestigador) as 'Cantidad
trabajos nivel 3',
dbo.fn_CantTrabajoPorNivel(4, i.idInvestigador) as 'Cantidad
trabajos nivel 4'
```

```

FROM Investigador i LEFT OUTER JOIN TAutores ta
ON i.idInvestigador = ta.idInvestigador
LEFT OUTER JOIN Trabajo t
ON ta.idTrab = t.idTrab
LEFT OUTER JOIN Lugares l
ON t.lugarPublic = l.idLugar
GO

```

/\* f - Para la universidad "ORT" mostrar el identificador de sus investigadores que tienen algún trabajo en el año actual y la cantidad de trabajos publicados en congresos de nivel 4. En caso de investigadores sin trabajos publicados en estos congresos pero con trabajos en proceso el año actual deben aparecer en el resultado.\*/

```

SELECT i.idInvestigador, (
    SELECT COUNT(DISTINCT tra.idTrab)
    FROM Lugares lu, TAutores x, Trabajo tra
    WHERE lu.idLugar = tra.lugarPublic
    AND i.idInvestigador = x.idInvestigador
    AND tra.idTrab = x.idTrab
    AND lu.nivelLugar = 4
    AND lu.tipoLugar = 'Congresos'
) AS 'Cant. de Trabajos en ORT para Nivel 4'
FROM Investigador i LEFT OUTER JOIN TAutores ta
ON i.idInvestigador = ta.idInvestigador
LEFT OUTER JOIN Trabajo t
ON ta.idTrab = t.idTrab
WHERE i.idUniversidad LIKE 'ORT'
AND YEAR(t.fechaInicio) = YEAR(GETDATE())
GROUP BY i.idInvestigador

```

```

/* g - Mostrar para cada universidad que tiene trabajos publicados,
los datos del último trabajo publicado.
Solucionar usando la función a).*/

```

```

SELECT DISTINCT U.nombre, dbo.fn_UltimoTrabajoPorUniv(l.universidad)
AS 'Datos de Trabajo publicado'
FROM Trabajo t, Lugares l, Universidad u
WHERE t.lugarPublic = l.idLugar
AND U.nombre = L.universidad

```

```

/* h- Eliminar las palabras claves no usadas en los trabajos.*/

```

```

DELETE FROM Tags
WHERE idTag NOT IN
(
    SELECT idtag
    FROM TTags
)
GO

```

```

/*#####*/
/*#####*/
/*#####*/
/*
                                SE PIDE #7
                                */
/*#####*/
/*#####*/
/*#####*/

```

```

/* a- Realizar una vista que muestre lista de Congresos y
para cada uno la cantidad de trabajos publicados que no tienen
autores que sean investigadores de la universidad anfitriona del
congreso.*/

```

```

CREATE VIEW View_ListaCongresos
AS
SELECT lu.nombre 'Congreso', COUNT(DISTINCT tra.idTrab) AS 'Cantidad
de trabajos'
FROM Lugares lu, Investigador inv, TAutores x, Trabajo tra
WHERE lu.idLugar = tra.lugarPublic
AND inv.idInvestigador = x.idInvestigador
AND tra.idTrab = x.idTrab

```

```

AND lu.tipoLugar LIKE 'Congresos'
AND tra.idTrab NOT IN (
    SELECT t.idTrab
    FROM Investigador i, Trabajo t, TAutores ta, Lugares l
    WHERE t.idTrab = ta.idTrab
    AND i.idInvestigador = ta.idInvestigador
    AND i.idUniversidad = lu.Universidad
)
GROUP BY lu.nombre
GO

```

```

select * from View_ListaCongresos

```

/\* b- Realizar una vista que muestre para cada Investigador, para cada tipo de trabajo la fecha de inicio del primer y último trabajo.  
 Todas los investigadores deben aparecer en el resultado, aunque no tengan trabajos que cumplan las condiciones. \*/

```

CREATE VIEW View_TrabajoInvestigadores
AS
SELECT i.idInvestigador Investigador, t.tipoTrab 'Tipo de trabajo',
MIN(t.fechaInicio) as 'Fecha Inicio primer Trabajo',
MAX(t.fechaInicio) as 'Fecha Inicio ultimo Trabajo'
FROM Investigador i LEFT OUTER JOIN TAutores ta
ON i.idInvestigador = ta.idInvestigador
LEFT OUTER JOIN Trabajo t
ON ta.idTrab = t.idTrab
GROUP BY i.idInvestigador, t.tipoTrab
GO

```

## Anexo con juego de datos a rechazar

```
/*#####*/
/*#####*/
/*#####*/
/*          SE PIDE #3      -      DATOS A RECHAZAR          */
/*#####*/
/*#####*/
/*#####*/
```

/\* 3. Ingreso de un juego completo de datos de prueba (será más valorada la calidad de los datos más que la cantidad. El mismo debería incluir ejemplos que deban ser rechazados por no cumplir con las restricciones implementadas.\*/

```
/*+++++*/
/*          TABLA UNIVERSIDAD          */
/*+++++*/
```

```
/* Caso a ser rechazado por contener PK duplicada*/
INSERT INTO Universidad
VALUES ('Udelar', 'Argentina', 'Buenos Aires', '44114444')
```

```
/* Caso a ser rechazado por pais = NULL*/
INSERT INTO Universidad
VALUES ('Nueva Universidad', NULL, 'Buenos Aires', '44554455')
```

```
/* Caso a ser rechazado por ciudad = NULL*/
INSERT INTO Universidad
VALUES ('Universidad del Nuevo Mundo', 'Uruguay', NULL, '55445544')
```

```
/* Caso a ser rechazado por telefono = NULL*/
INSERT INTO Universidad (nombre, pais, ciudad)
VALUES ('Universidad Cornell', 'Uruguay', 'Montevideo')
```

```
/*+++++*/
/*          TABLA LUGARES          */
/*+++++*/
```

```

/* Caso a ser rechazado por idLugar duplicado */
INSERT INTO Lugares
VALUES(1, 'Hotel Guadalajara', 4, 2017, 11, 8, null, '', 'Udelar',
'Revistas')

/* Caso a ser rechazado por nombre = null */
INSERT INTO Lugares (idLugar, nombre, nivelLugar, año, mes, diaIni,
diaFin, link, universidad, tipoLugar)
VALUES(6, null, 4, 2017, 11, 8, null, '', 'Udelar', 'Revistas')

/* Caso a ser rechazado por nivel > 4 */
INSERT INTO Lugares
VALUES(6, 'Hotel Guadalajara', 8, 2017, 11, 8, null, '', 'Udelar',
'Revistas')

/* Caso a ser rechazado por año > año actual */
INSERT INTO Lugares
VALUES(6, 'Hotel Guadalajara', 4, 2300, 11, 8, null, '', 'Udelar',
'Revistas')

/* Caso a ser rechazado por mes > 12 */
INSERT INTO Lugares
VALUES(6, 'Hotel Guadalajara', 4, 2015, 98, 8, null, '', 'Udelar',
'Revistas')

/* Caso a ser rechazado por mes < 1 */
INSERT INTO Lugares
VALUES(6, 'Hotel Guadalajara', 4, 2015, 0, 8, null, '', 'Udelar',
'Revistas')

/* Caso a ser rechazado por dia > 31 */
INSERT INTO Lugares
VALUES(6, 'Hotel Guadalajara', 4, 2015, 11, 50, null, '', 'Udelar',
'Revistas')

/* Caso a ser rechazado por tipoLugar no entre los permtdos */
INSERT INTO Lugares
VALUES(6, 'Hotel Guadalajara', 4, 2015, 11, 20, null, '', 'Udelar',
'Verduleria')

/* Caso a ser rechazado por nombre repetido */
INSERT INTO Lugares

```

```
VALUES(6, 'Hotel Dazzler', 4, 2015, 11, 20, null, '', 'Udelar',  
'Revistas')
```

```
/* Caso a ser rechazado por universidad = null */  
INSERT INTO Lugares (idLugar, nombre, nivelLugar, año, mes, diaIni,  
diaFin, link, universidad, tipoLugar)  
VALUES(6, 'Hotel Guadalajara', 4, 2016, 11, 8, null, '', null,  
'Revistas')
```

```
/*+++++*/  
/*                                TABLA INVESTIGADOR                                */  
/*+++++*/
```

```
/*Datos a rechazar*/  
/* Caso a ser rechazado por ingresar idInvestigador */  
INSERT INTO Investigador (idInvestigador,nombre, mail, telefono,  
carrera, nivelInvestig,cantTrabPub,idUniversidad)  
VALUES (365,'Marcio Avellanal', 'mavellanal@investigadores.com.uy',  
'45129685', 'Licenciatura en Matemáticas', 'EDoctor', 5,'Universidad  
Federal de Alagoas')
```

```
/* Caso a ser rechazado por ingresar Universidad inexistente */  
INSERT INTO Investigador (nombre, mail, telefono, carrera,  
nivelInvestig,cantTrabPub,idUniversidad)  
VALUES ('Marcio Avellanal', 'mavellanal@investigadores.com.uy',  
'45129685', 'Licenciatura en Matemáticas', 'EDoctor', 5,'Universidad  
de Michigan')
```

```
/* Caso a ser rechazado por no ingresar nombre*/  
INSERT INTO Investigador (mail, telefono, carrera,  
nivelInvestig,cantTrabPub,idUniversidad)  
VALUES ('mavellanal@investigadores.com.uy', '45129685',  
'Licenciatura en Matemáticas', 'EDoctor', 5,'Universidad de  
Michigan')
```

```
/* Caso a ser rechazado por no ingresar nivelInvestig*/  
INSERT INTO Investigador (nombre, mail, telefono,  
carrera,cantTrabPub,idUniversidad)  
VALUES ('Marcio Avellanal', 'mavellanal@investigadores.com.uy',  
'45129685', 'Licenciatura en Matemáticas', 5,'Universidad de  
Michigan')
```



```

/* Caso a ser rechazado por no ingresar cantTrabPub*/
INSERT INTO Investigador (nombre, mail, telefono, carrera,
nivelInvestig,idUniversidad)
VALUES ('Marcio Avellanal', 'mavellanal@investigadores.com.uy',
'45129685', 'Licenciatura en Matemáticas', 'EDoctor','Universidad de
Michigan')

```

```

/* Caso a ser rechazado por no ingresar mail unico*/
INSERT INTO Investigador (nombre, mail, telefono, carrera,
nivelInvestig,cantTrabPub,idUniversidad)
VALUES ('Marcio Avellanal', 'mavellanal@investigadores.com.uy',
'98765432', 'Dcotor en Medicina', 'EDoctor', 4,'Udelar')

```

```

/*+++++*/
/*                               Tabla TRABAJO                               */
/*+++++*/

```

```

/*Datos a rechazar*/
/* Caso a ser rechazado por Descripcion > 200 Caracteres */
INSERT INTO Trabajo
VALUES('Investigacion de como hacer las cosas mal', 'Nor hence hoped
her after other known defer his. For county now sister engage had
season better had waited. Occasional mrs interested far expression
acceptance. Day either mrs talent pulled men rather 201',
'articulo', '2016-04-03', null,1,'P3')

```

```

/* Caso a ser rechazado por tipoTrab no permitido */
INSERT INTO Trabajo
VALUES('Investigacion sobre el mal hacer', 'Investigacion sobre
cuando las cosas se macen hal', 'resumen', '2016-04-03',
'https://www.google.com.uy',1,'P4')

```

```

/*+++++*/
/*                               Tabla TAGS                               */
/*+++++*/

```

```

/* Caso a ser rechazado por palabra NULL */
INSERT INTO Tags(palabra)

```

VALUES (NULL)

```
/*+++++*/
/*                               Tabla TTAGS                               */
/*+++++*/
```

```
/*Datos a rechazar*/
/*Caso a ser rechazado por PK duplicado*/
INSERT INTO TTags
VALUES ('A0',1)
```

```
/*Caso a ser rechazado por IdTag inexistente*/
INSERT INTO TTags
VALUES ('A0',98)
```

```
/*Caso a ser rechazado por IdTrab inexistente*/
INSERT INTO TTags
VALUES ('Z15',27)
```

```
/*Caso a ser rechazado por IdTrab null*/
INSERT INTO TTags(idTag)
VALUES (27)
```

```
/*+++++*/
/*                               Tabla TAutores                               */
/*+++++*/
```

```
/*Datos a rechazar*/
/*Caso a ser rechazado por PK duplicada*/
INSERT INTO TAutores
VALUES('A0',1,'autor-ppal')
```

```
/*Caso a ser rechazado por IdTrab null*/
INSERT INTO TAutores(idInvestigador,rolinvestig)
VALUES(8,'autor-director')
```

```
/*Caso a ser rechazado por IdInvestigador null*/
INSERT INTO TAutores(idTrab,rolinvestig)
VALUES('A1','autor-director')
```

```
/*Caso a ser rechazado por rol fuera de rango*/
```

```
INSERT INTO TAutores(idTrab,idInvestigador,rolinvestig)
VALUES('A1',7,'editor')
```

```
/*+++++*/
/*                               Tabla REFERENCIAS                               */
/*+++++*/
```

```
/*Datos a rechazar*/
```

```
/* Caso a ser rechazado por: idTrab = null */
```

```
INSERT INTO Referencias
```

```
VALUES(null,'A1')
```

```
/* Caso a ser rechazado por: idTrabReferenciado = null */
```

```
INSERT INTO Referencias
```

```
VALUES('02',null)
```

```
GO
```

```
/* Caso a ser rechazado por: se referencia asimismo */
```

```
INSERT INTO Referencias
```

```
VALUES('02','02')
```

# Impresión de los scripts

## Scripts de restricciones de integridad

### Restricciones para UNIVERSIDAD

```
/* UNIVERSIDAD */

ALTER TABLE Universidad
  ALTER COLUMN nombre VARCHAR(100) NOT NULL;
GO

ALTER TABLE Universidad
  ADD CONSTRAINT pk_nombre Primary key(nombre)
GO

ALTER TABLE Universidad
  ADD telefono varchar(20) not null;
GO

/*-----*/
```

100 %

Mensajes

Comandos completados correctamente.

### Restricciones para INVESTIGADOR

```
/* INVESTIGADOR */

ALTER TABLE Investigador
  DROP COLUMN idInvestigador
GO

ALTER TABLE Investigador
  ADD idInvestigador INT NOT NULL IDENTITY(1,1)
GO

ALTER TABLE Investigador
  ADD idUniversidad VARCHAR(100) NOT NULL
GO

ALTER TABLE Investigador
  ADD CONSTRAINT Investigador_PK PRIMARY KEY (idInvestigador)
GO

ALTER TABLE Investigador
  ADD CONSTRAINT Investigador_FK FOREIGN KEY (idUniversidad)
  REFERENCES Universidad
GO

ALTER TABLE Investigador
  ADD CONSTRAINT NivelInv_CH CHECK (nivelInvestig IN ('EGrado', 'EMaestria', 'EDoctor', 'Doctor'))
GO

ALTER TABLE Investigador
  ADD UNIQUE (Mail)
GO

ALTER TABLE Investigador
  ALTER COLUMN cantTrabPub INT NOT NULL
```

83 %

Mensajes

Comandos completados correctamente.

```

/*-----*/
/*Disparador para controlar que un INVESTIGADOR no cambie de UNIVERSIDAD */
CREATE TRIGGER INVESTIGADOR_UNIVERSIDAD
ON Investigador
INSTEAD OF UPDATE
AS
BEGIN
    IF(EXISTS
        (
            SELECT *
            FROM Investigador x, inserted i
            WHERE x.idInvestigador = i.idInvestigador
            AND x.idUniversidad <> i.idUniversidad
        )
    )
    BEGIN
        PRINT 'No se admiten cambios de UNIVERSIDAD para un INVESTIGADOR.'
    END
    ELSE
    BEGIN
        UPDATE Investigador
        SET nombre = i.nombre, mail = i.mail, telefono = i.telefono,
        carrera = i.carrera, nivelInvestig = i.nivelInvestig,
        cantTrabPub = i.cantTrabPub, idUniversidad = i.idUniversidad
        FROM inserted i, Investigador inv
        where i.idInvestigador = inv.idInvestigador
    END
END
GO

```

91 %

Mensajes

Comandos completados correctamente.

## Restricciones para TRABAJO

```

/* TRABAJO */
ALTER TABLE Trabajo
DROP COLUMN idTrab
GO

ALTER TABLE Trabajo
ADD idTrab varchar(10) not null;
GO

ALTER TABLE Trabajo
ALTER COLUMN descripTrab VARCHAR(200)
GO

ALTER TABLE Trabajo
ADD CONSTRAINT tipoTrab_check CHECK (tipoTrab IN ('poster', 'articulo', 'capitulo', 'otro'))
GO

ALTER TABLE Trabajo
ADD CONSTRAINT Trabajo_PK PRIMARY KEY (idTrab)
GO

ALTER TABLE Trabajo
ADD CONSTRAINT Trabajo_FK FOREIGN KEY (lugarPublic)
REFERENCES Lugares
GO

/*-----*/

```

100 %

Mensajes

Comandos completados correctamente.

100 %

```

/*-----*/
/*Disparador para generar ID Trabajo */

CREATE TRIGGER trig_idTrab
ON Trabajo
INSTEAD OF INSERT
AS
BEGIN
    IF( NOT EXISTS( Select COUNT(*) From inserted GROUP BY nomTrab having count(*) > 1))
    BEGIN
        DECLARE @ultINS int;
        SET @ultINS = (select COUNT(*) from Trabajo where tipoTrab in (select tipoTrab from inserted));

        DECLARE @alphaNumID varchar(10);
        SELECT @alphaNumID = UPPER(SUBSTRING(tipoTrab, 1, 1)) from inserted;

        SET @alphaNumID = @alphaNumID + CONVERT(varchar(10), @ultINS);

        INSERT INTO Trabajo
        SELECT nomTrab, descripTrab, tipoTrab, fechaInicio, linkTrab, lugarPublic, @alphaNumID
        FROM inserted
    END
    ELSE
    BEGIN
        PRINT 'No se admiten inserciones múltiples.'
    END
END
GO

```

Mensajes

Comandos completados correctamente.

## Restricciones para TAGS / TTAGS

```

/* TAGS */
ALTER TABLE Tags
DROP COLUMN idTag
GO

ALTER TABLE Tags
ADD idTag INT IDENTITY(1,2) NOT NULL
GO

ALTER TABLE Tags
ADD CONSTRAINT Tags_PK PRIMARY KEY (idTag)
GO

/*-----*/
/* TTAGS */
ALTER TABLE TTags
ADD CONSTRAINT TTags_PK PRIMARY KEY (idTrab, idTag)
GO

ALTER TABLE TTags
ADD CONSTRAINT TTags_FK1 FOREIGN KEY (idTrab)
REFERENCES Trabajo
GO

ALTER TABLE TTags
ADD CONSTRAINT TTags_FK2 FOREIGN KEY (idTag)
REFERENCES Tags
GO

```

Mensajes

Comandos completados correctamente.

## Restricciones para TAUtores

```
/*-----*/
/* TAUtores */
ALTER TABLE TAUtores
ADD CONSTRAINT TAUtores_PK PRIMARY KEY (idTrab, idInvestigador)
GO

ALTER TABLE TAUtores
ADD CONSTRAINT TAUtores_FK1 FOREIGN KEY (idTrab)
REFERENCES Trabajo
GO

ALTER TABLE TAUtores
ADD CONSTRAINT TAUtores_FK2 FOREIGN KEY (idInvestigador)
REFERENCES Investigador
GO

ALTER TABLE TAUtores
ALTER COLUMN rolinvestig varchar(20);
GO

ALTER TABLE TAUtores
ADD CONSTRAINT rolinvestig_check CHECK (rolinvestig IN ('autor-ppal', 'autor-sec', 'autor-director'))
GO
```

91 % < >

Mensajes

Comandos completados correctamente.

100 % < >

## Restricciones para REFERENCIAS

```
/* REFERENCIAS */
ALTER TABLE Referencias
ADD CONSTRAINT Referencias_PK PRIMARY KEY (idTrab, idTrabReferenciado)
GO

ALTER TABLE Referencias
ADD CONSTRAINT Referencias_FK_Trab FOREIGN KEY (idTrab)
REFERENCES Trabajo
GO

ALTER TABLE Referencias
ADD CONSTRAINT Referencias_FK_TrabRef FOREIGN KEY (idTrabReferenciado)
REFERENCES Trabajo
GO

ALTER TABLE Referencias
ADD CONSTRAINT Referencias_Referencia CHECK (idTrab <> idTrabReferenciado)
```

91 % < >

Mensajes

Comandos completados correctamente.

100 % < >

## Restricciones para LUGARES

```
/* LUGARES */
ALTER TABLE Lugares
ALTER COLUMN nombre varchar(250) not null;
GO

ALTER TABLE Lugares
ADD CONSTRAINT nombre_uniq unique (nombre);
GO

ALTER TABLE Lugares
ADD tipoLugar varchar(10) not null;
GO

ALTER TABLE Lugares
ADD CONSTRAINT tipoLugar_check CHECK (tipoLugar IN ('Congresos', 'Revistas', 'Libros'))
GO

ALTER TABLE Lugares
ADD CONSTRAINT nivellugar_check CHECK (nivellugar BETWEEN 1 and 4);
GO

ALTER TABLE Lugares
ALTER COLUMN universidad VARCHAR(100) NOT NULL
GO

ALTER TABLE Lugares
ADD CONSTRAINT Lugares_FK FOREIGN KEY (universidad)
REFERENCES Universidad
GO
```

91 % < >

Mensajes

Comandos completados correctamente.

100 % < >

```
ALTER TABLE Lugares
ADD CONSTRAINT mes_check CHECK (mes BETWEEN 1 and 12)
GO

ALTER TABLE Lugares
ADD CONSTRAINT diaI_check CHECK (diaIni BETWEEN 1 and 31)
GO

ALTER TABLE Lugares
ADD CONSTRAINT diaF_check CHECK (diaFin BETWEEN 1 and 31)
GO

ALTER TABLE Lugares
ADD CONSTRAINT año_check CHECK (año BETWEEN 1900 and YEAR( GETDATE()));
GO

/*-----*/
```

91 % < >

Mensajes

Comandos completados correctamente.

100 % < >



```
/*-----*/
/* Disparador para controlar que diaFin no sea nulo cuando tipolugar tiene
valor 'Congresos' en tabla LUGARES*/

CREATE TRIGGER EvitarDiaFinNulo_LUGARES
ON Lugares
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @anio VARCHAR (4),
            @mes VARCHAR (2),
            @diaIni VARCHAR (2),
            @diaFin VARCHAR (10),
            @fechaInicio DATE,
            @fechaFin DATE,
            @fechaActual DATE

    SELECT @anio = CAST(año AS VARCHAR(4)) FROM inserted
    SELECT @mes = CAST(mes AS VARCHAR(2)) FROM inserted
    SELECT @diaIni = CAST(diaIni AS VARCHAR(2)) FROM inserted
    SET @fechaInicio = CONVERT(date, @anio+'-'+@mes+'-'+@diaIni)

    IF(EXISTS (SELECT tipolugar from inserted WHERE tipolugar NOT LIKE 'Congresos')
    AND EXISTS(SELECT diaFin FROM inserted WHERE diafin IS NULL AND tipolugar NOT LIKE 'Congresos'))
    BEGIN
        INSERT Lugares
        SELECT idLugar = inserted.idLugar, nombre = inserted.nombre, nivellugar = inserted.nivellugar,
            año = inserted.año, mes = inserted.mes, diaIni = inserted.diaIni, diaFin = null,
            link = inserted.link, universidad = inserted.universidad, tipolugar = inserted.tipolugar
        FROM inserted
    END

    ELSE
    IF(EXISTS (SELECT tipolugar from inserted WHERE tipolugar LIKE 'Congresos')
    AND NOT EXISTS (SELECT diaFin FROM inserted WHERE diafin IS NULL AND tipolugar LIKE 'Congresos'))
    BEGIN
        SELECT @diaFin = CAST(diaFin AS VARCHAR(2)) FROM inserted
        SET @fechaFin = CONVERT(date, @anio+'-'+@mes+'-'+@diaFin)
        SET @fechaActual = GETDATE()
        IF(@fechaInicio < @fechaFin AND @fechaInicio < @fechaActual)
        BEGIN
            INSERT Lugares
            SELECT idLugar = inserted.idLugar, nombre = inserted.nombre, nivellugar = inserted.nivellugar,
                año = inserted.año, mes = inserted.mes, diaIni = inserted.diaIni, diaFin = inserted.diaFin,
                link = inserted.link, universidad = inserted.universidad, tipolugar = inserted.tipolugar
            FROM inserted
        END
        ELSE
        BEGIN
            PRINT 'La fecha de inicio debe ser anterior a la fecha fin y ambas deben ser anteriores a la fecha actual.'
        END
    END
    ELSE
    BEGIN
        PRINT 'Todos los congresos deben tener un día fin.'
    END
END
GO
```

83 %

Mensajes

Comandos completados correctamente.

100 %

## Scripts creación Índices

```
CREATE INDEX i_investigador_uni ON Investigador(idUniversidad);
CREATE INDEX i_lugares_uni ON Lugares(universidad);
CREATE INDEX i_autores_investigador ON TAutores(idInvestigador);
CREATE INDEX i_trabajo_lugar ON Trabajo(lugarPublic);
CREATE INDEX i_trabajo_referenciado ON Referencias(idTrabReferenciado);
CREATE INDEX i_ttags_tags ON TTags(idTag);
GO
```

83 %

Mensajes

Comandos completados correctamente.

100 %

## Scripts de ingreso de datos de prueba

### Universidad

```
/*+++++ Tabla UNIVERSIDAD +++++*/
/*
 Tabla UNIVERSIDAD
*/
/*+++++*/

/*Datos OK*/
INSERT INTO Universidad
VALUES
('Udelar', 'Uruguay', 'Montevideo', '29009999'),
('ORT', 'Uruguay', 'Montevideo', '27007777'),
('UCUDAL', 'Uruguay', 'Montevideo', '24004444'),
('UM', 'Uruguay', 'Montevideo', '26006666'),
('Universidad de Palermo', 'Argentina', 'Buenos Aires', '44116666'),
('UBA', 'Argentina', 'Buenos Aires', '44110000'),
('Universidad de Córdoba', 'Argentina', 'Córdoba', '45000000'),
('Universidad de Brasilia', 'Brasil', 'Brasilia', '62501253'),
('Universidad Federal de Alagoas', 'Brasil', 'Alagoas', '78514569'),
('Universidad de Amazonas', 'Brasil', 'Amazonas', '35358978')
```

83 %

Mensajes

(10 filas afectadas)

100 %

### Lugares

```
/*+++++ Tabla LUGARES +++++*/
/*
 Tabla LUGARES
*/
/*+++++*/

/* Datos OK */
INSERT INTO Lugares
VALUES
(1, 'Teatro Solis', 4, 2016, 11, 8, null, 'http://www.teatrosolis.org.uy', 'Udelar', 'Revistas'),
(2, 'LATU', 4, 2015, 11, 9, 13, 'www.latu.org.uy', 'ORT', 'Congresos'),
(3, 'Radisson Victoria Plaza Hotel', 4, 2015, 5, 20, null, 'https://www.radissonblu.com', 'UM', 'Libros'),
(4, 'Holiday Inn', 2, 2017, 2, 10, 16, 'https://www.ihg.com', 'UCUDAL', 'Congresos'),
(5, 'Hotel Dazzler', 1, 2017, 8, 8, null, 'https://www.dazzlerhoteles.com', 'UBA', 'Revistas')
```

83 %

Mensajes

(5 filas afectadas)

100 %

## Investigador

```
/*++++++*/
/*                               Tabla INVESTIGADOR                               */
/*++++++*/

/*Datos OK*/
INSERT INTO Investigador (nombre, mail, telefono, carrera, nivelInvestig,cantTrabPub,idUniversidad)
VALUES
('Marcelo López', 'mlopez@investigadores.com.uy', '098999999', 'Ingeniería Química', 'Egrado', 5, 'Udelar'),
('Laura Marquisio', 'lmarqui@investigadores.com.uy', '098111111', 'Licenciatura en Relaciones Internacionales', 'EMaestria', 1, 'Udelar'),
('Fabián Méndez', 'fmendez@investigadores.com.uy', '095951135', 'Licenciatura en Economía', 'EMaestria', 3, 'Udelar'),
('Silvia Luque', 'sluque@investigadores.com.uy', '096457215', 'Licenciatura en Economía', 'EDoctor', 9, 'Udelar'),
('Silvio Duarte', 'sduarte@investigadores.com.uy', '099485245', 'Licenciatura en Sistemas', 'Doctor', 15, 'Udelar'),
('Walter Clitish', 'wclitish@investigadores.com.uy', '099123456', 'Licenciatura en Letras', 'EDoctor', 12, 'Udelar'),
('Maicol Uriarte', 'muriarte@investigadores.com.uy', '15648524', 'Licenciatura en Bellas Artes', 'EMaestria', 9, 'UBA'),
('Mariana Ifrán', 'mifran@investigadores.com.uy', '15677425', 'Ingeniería Naval', 'Egrado', 1, 'Universidad de Córdoba'),
('Linda Cibils', 'lcibils@investigadores.com.uy', '48579568', 'Licenciatura en Ciencias Biológicas', 'EDoctor', 5, 'Universidad de Córdoba'),
('Marcio Avellanal', 'mavellanal@investigadores.com.uy', '45129685', 'Licenciatura en Matemáticas', 'EDoctor', 5, 'Universidad de Córdoba'),
('Guillermo Polachek', 'polachek@ort.edu.uy', '091111111', 'Ingeniería', 'Doctor', 50, 'ORT'),
('Sebastian Villar', 'villar@uamazonas.edu.uy', '092222222', 'Ingeniería', 'Doctor', 50, 'Universidad de Amazonas'),
('Atuagualpo Galpones', 'atagua@gmail.com', '093333333', 'Ingeniería', 'Doctor', 3, 'ORT')
/*++++++*/
```

83 % < >

Mensajes

(13 filas afectadas)

100 % < >

## Trabajo

```
/*+++++*/
/*                               Tabla TRABAJO                               */
/*+++++*/
/* Datos OK */
INSERT INTO Trabajo
VALUES
('Investigacion GARZA CUCA', 'La garza cuca o también denominada garza mora (Ardea cocoi) es un ave nativa del Centro y Sudamérica, se estudia su ambiente y entorno',
'articulo', '2016-04-03', 'https://www.infoanimales.com/informacion-sobre-la-garza-cuca',1, 'id')

INSERT INTO Trabajo
VALUES
('Venado de campo', 'Investigacion sobre el venado de campo, uno de los integrantes más característicos de la fauna uruguaya',
'capitulo', '2017-02-01', 'http://blogs.ceibal.edu.uy/formacion/coleccion-de-recursos/venado-de-campo/',2, 'id')

INSERT INTO Trabajo
VALUES
('Investigacion sobre el Agua', 'El agua es un bien y un recurso cada vez mas escaso, que debe ser valorado, protegido y recuperado',
'poster', '2017-05-17', 'https://es.slideshare.net/sssanchezayelen/investigacin-sobre-el-agua',3, 'id')
```

83 %

Mensajes

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

68 %

```
INSERT INTO Trabajo
VALUES
('Investigacion sobre medio ambiente ', 'El análisis de lo ambiental desde la perspectiva de lo social',
'Otro', '2017-08-20', 'http://cis.ufro.cl/index.php?option=com_content&view=article&id=45&Itemid=34',5, 'id')

INSERT INTO Trabajo
VALUES
('Investigacion sobre las drogas', 'La drogadicción es una enfermedad que consiste en la dependencia de sustancias que afectan el sistema nervioso central y las funciones cerebrales',
'articulo', '2017-05-17', 'https://www.monografias.com/docs/Investigacion-sobre-las-drogas-FKJQBHKYVZ',4, 'id')

INSERT INTO Trabajo
VALUES
('Investigacion sobre Cultura maya ', 'La civilización maya es sin duda la más fascinante de las antiguas culturas americanas',
'Otro', '2017-04-28', 'https://www.biografiasyvidas.com/historia/cultura_maya.htm',5, 'id')

INSERT INTO Trabajo
VALUES
('Investigacion sobre SQL', 'Lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales', 'articulo', '2017-12-08', 'https://es.wikipedia.org/wiki/SQL',5, 'id')
```

83 %

Mensajes

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

68 %



```
INSERT INTO Trabajo
VALUES
('Investigacion sobre XQuery', 'XQuery es un lenguaje de consulta para fuentes de datos XML',
'articulo', '2017-12-07', 'https://es.wikipedia.org/wiki/XQuery',5, 'id')

INSERT INTO Trabajo
VALUES
('Investigacion sobre Bases de Datos Oracle', 'Oracle Database es un sistema de gestión de
base de datos de tipo objeto-relacional',
'articulo', '2017-12-06', 'https://es.wikipedia.org/wiki/Oracle_Database',5, 'id')

INSERT INTO Trabajo
VALUES
('Investigacion sobre Bases de Datos', 'Una base de datos o banco de datos es un conjunto de datos pertenecientes
a un mismo contexto y almacenados sistemáticamente para su posterior uso',
'capitulo', '2017-12-08', 'https://es.wikipedia.org/wiki/Base_de_datos',5, 'id')
|

83 %
```

Mensajes

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

83 %

```
INSERT INTO Trabajo
VALUES
('Investigacion sobre Windows', 'Microsoft Windows es un sistema operativo, es decir,
un conjunto de programas que posibilita la administración de los recursos de una computadora',
'articulo', '2017-11-10', 'https://definicion.de/windows/',4, 'id')

INSERT INTO Trabajo
VALUES
('Investigacion sobre SO. Ubuntu', 'Ubuntu es una distribución del sistema operativo GNU/Linux
y que se distribuye como software libre',
'articulo', '2017-11-09', 'https://es.wikipedia.org/wiki/Ubuntu',4, 'id')

INSERT INTO Trabajo
VALUES
('Investigacion sobre Sistemas Operativos', 'Un sistema operativo es el software principal o conjunto de programas
de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación
de software', 'articulo', '2017-11-08', 'https://es.wikipedia.org/wiki/Sistema_operativo',4, 'id')

91 %
```

Mensajes

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

83 %

## Tags

```
/*+++++ Tabla TAGS +++++*/
/*
/*+++++ Tabla TAGS +++++*/

/*Datos OK*/
INSERT INTO Tags
VALUES
('garza'),
('cuca'),
('venado'),
('campo'),
('fauna'),
('animales'),
('silvestre'),
('agua'),
('ambiente'),
('ecologia'),
('drogas'),
('adicciones'),
('cultura'),|
('mayas'),
('BASE DE DATOS'),
('sql'),
('xquery'),
('oracle'),
('windows'),
('ubuntu'),
('sistema-operativo')
```

91 % < >

Mensajes

(21 filas afectadas)

100 % < >

## TTags

```
/*+++++ Tabla TTAGS +++++*/
/*
/*+++++ Tabla TTAGS +++++*/

/*Datos OK*/
INSERT INTO TTags
VALUES
('A0',1),
('A0',3),
('C0',5),
('O0',7),
('O1',9),
('P0',15),
('O1',27),
('A2',29),
('A3',29),
('A4',29),
('C1',29),
('A2',31),
('A3',33),
('A4',35),
('A5',41),
('A6',41),
('A7',41),
('A5',37),
('A6',39)
```

91 % < >

Mensajes

(19 filas afectadas)

100 % < >

## Tautores

```
/*+++++++Tabla TAUTORES+++++*/
/*                               */
/*+++++++*/

INSERT INTO Tautores
VALUES
('A0',1,'autor-ppal'),
('A0',5,'autor-sec'),
('A6',13,'autor-sec'),
('C1',13,'autor-sec'),
('C0',2,'autor-director'),
('O1',2,'autor-ppal'),
('P0',3,'autor-director'),
('A1',4,'autor-ppal'),
('A1',6,'autor-ppal'),
('A1',9,'autor-sec'),
('A0',9,'autor-sec'),
('A1',8,'autor-director'),
('A2',11,'autor-ppal'),
('A2',12,'autor-ppal'),
('A3',11,'autor-ppal'),
('A3',12,'autor-ppal'),
('A4',11,'autor-ppal'),
('A4',12,'autor-ppal'),
('C1',11,'autor-ppal'),
('C1',12,'autor-ppal'),
('A5',11,'autor-ppal'),
('A5',12,'autor-ppal'),
('A6',11,'autor-ppal'),
('A6',12,'autor-ppal'),
('A7',11,'autor-ppal'),
('A7',12,'autor-ppal')

83 % < >
Mensajes
(26 filas afectadas)
100 % < >
```

## Referencias

```
/*+++++++Tabla REFERENCIAS+++++*/
/*                               */
/*+++++++*/
/* Datos OK */

INSERT INTO Referencias
VALUES('A0','C0')
INSERT INTO Referencias
VALUES('P0','O0')
INSERT INTO Referencias
VALUES('A0','A1')
INSERT INTO Referencias
VALUES('C1','A2')
INSERT INTO Referencias
VALUES('C1','A3')
INSERT INTO Referencias
VALUES('C1','A4')
INSERT INTO Referencias
VALUES('A7','A6')
INSERT INTO Referencias
VALUES('A7','A5')

91 % < >
Mensajes
(1 filas afectadas)
(1 filas afectadas)
(1 filas afectadas)
(1 filas afectadas)
(1 filas afectadas)
(1 filas afectadas)
(1 filas afectadas)
(1 filas afectadas)
62 % < >
```

## Script con la creación de funciones y procedimientos

4a - Crear una función que dada una universidad devuelva el último trabajo publicado por esta. Si hay más de uno devolver uno cualquiera.

```
CREATE FUNCTION fn_UltimoTrabajoPorUniv (@unaUniversidad VARCHAR(100))
RETURNS VARCHAR(110)
AS
BEGIN
    DECLARE @ultTrabajo VARCHAR(10)
    DECLARE @nomTrabajo VARCHAR(100)
    DECLARE @ret VARCHAR(110)

    SELECT @ultTrabajo = idTrab , @nomTrabajo = nomTrab
    FROM Trabajo tr
    WHERE lugarPublic IN
    (
        SELECT idLugar
        FROM Lugares
        WHERE universidad = @unaUniversidad
    ) AND tr.fechaInicio IN (Select MAX(fechaInicio) from Trabajo WHERE lugarPublic IN
    (
        SELECT idLugar
        FROM Lugares
        WHERE universidad = @unaUniversidad
    ))

    set @ret = @ultTrabajo + ' - ' + @nomTrabajo;

    RETURN @ret
END
GO

declare @ultTrabUni VARCHAR(110);
set @ultTrabUni = dbo.fn_UltimoTrabajoPorUniv('UBA')
PRINT @ultTrabUni
GO
```

83 %

Mensajes

C1 - Investigación sobre Bases de Datos

100 %



4b - Crear una función almacenada que reciba como parámetro un trabajo y devuelva la cantidad de referencias externas que tiene.

```
CREATE FUNCTION fn_CantReferenciasExt ( @unTrabajo VARCHAR(100))
RETURNS INT
AS
BEGIN
    DECLARE @cantReferencias INT
    IF(EXISTS ( SELECT * FROM Referencias r WHERE r.idTrab = @unTrabajo))
    BEGIN
        SELECT @cantReferencias = COUNT(*)
        FROM Referencias r
        WHERE r.idTrab = @unTrabajo
        AND r.idTrabReferenciado NOT IN(
            SELECT idTrab
            from TAutores a
            where idInvestigador in (
                select idInvestigador
                from TAutores b
                where r.idTrab = b.idTrab
            )
        )
    END
    ELSE
    BEGIN
        SET @cantReferencias = 0
    END
    RETURN @cantReferencias
END
GO

declare @cantRef int;
set @cantRef = dbo.fn_CantReferenciasExt('A0')
PRINT @cantRef
```

83 %

Mensajes

1

100 %

4c - Crear una función que reciba dos investigadores y devuelva la cantidad de trabajos publicados en los cuales ambos investigadores fueron autores y alguno de los dos o los dos fueron autores principales

```
/*+++++*/  
/* 4c - Crear una función que reciba dos investigadores y devuelva  
la cantidad de trabajos publicados en los cuales ambos investigadores fueron autores  
y alguno de los dos o los dos fueron autores principales.*/  
  
CREATE FUNCTION fn_CantTrabPublicados  
(  
    @investigadoraA INT,  
    @investigadorB INT  
)  
RETURNS INT  
AS  
BEGIN  
    DECLARE @cantTrabajos INT  
  
    SELECT @cantTrabajos = COUNT (DISTINCT x.idTrab)  
    from TAutores x, TAutores y  
    where x.idInvestigador = @investigadoraA  
    and y.idInvestigador = @investigadorB  
    and x.idTrab = y.idTrab  
    and (x.rolinvestig = 'autor-ppal' or y.rolinvestig = 'autor-ppal')  
  
    RETURN @cantTrabajos  
END  
GO  
  
declare @cantTrabInvs int;  
set @cantTrabInvs = dbo.fn_CantTrabPublicados(11,12)  
PRINT @cantTrabInvs
```

83 %

Mensajes

7

100 %

4d - Crear una función o procedimiento, según considere/corresponda, que dado un investigador actualice el campo cantidad de trabajos publicados registrados de la tabla Investigador, y devuelva una indicación de si la cantidad que estaba registrada (antes de la actualización) era correcta o no.

```
CREATE PROCEDURE spu_UpdateCantTrab
@unInvestigador INT,
@mensaje VARCHAR(200) output
AS
BEGIN
    DECLARE @cantTrabajosEfectivos INT,
            @cantTrabajosContados INT

    SELECT @cantTrabajosEfectivos = COUNT(*)
    FROM TAutores
    WHERE idInvestigador = @unInvestigador

    SELECT @cantTrabajosContados = cantTrabPub
    FROM Investigador
    WHERE idInvestigador = @unInvestigador

    IF(@cantTrabajosContados = @cantTrabajosEfectivos)
    BEGIN
        SET @mensaje = 'La cantidad de trabajos era correcta, no fue necesario actualizar la tabla.'
    END
    ELSE
    BEGIN
        UPDATE Investigador
        SET cantTrabPub = @cantTrabajosEfectivos
        WHERE idInvestigador = @unInvestigador
        SET @mensaje = 'La cantidad de trabajos no era correcta, la tabla se actualizó con el valor correcto.'
    END
END
GO

DECLARE @mensaje varchar(200)
EXEC spu_UpdateCantTrab 9, @mensaje output
PRINT @mensaje
```

91 %

Mensajes

(1 filas afectadas)

(1 filas afectadas)

La cantidad de trabajos no era correcta, la tabla se actualizó con el valor correcto.

100 %

4.e Crear una función que dado dos años (año1 y año2) y una palabra clave devuelva la cantidad de trabajos publicados en libros o revistas en el rango de años y que trata el tema indicado por la palabra clave.

```
CREATE FUNCTION fn_CantTrbJAniosClave(
    @anio1 INT,
    @anio2 INT,
    @clave varchar(50)
)
RETURNS INT
AS
BEGIN
    Declare @cantidad int;
    select @cantidad = COUNT(*)
    FROM Trabajo
    WHERE idTrab in (
        SELECT idTrab
        From TTags
        where idTag in(
            select idTag
            from Tags
            where palabra = @clave
        )
    ) AND lugarPublic in (
        SELECT idLugar
        from Lugares
        where (tipoLugar = 'Revistas' or tipoLugar = 'Libros')
        AND año BETWEEN @anio1 AND @anio2
    )
    RETURN @cantidad
END
GO
```

Mensajes

Comandos completados correctamente.

```
declare @cantTrabInvs int;
set @cantTrabInvs = dbo.fn_CantTrbJAniosClave(2015, 2016, 'garza')
PRINT @cantTrabInvs
```

Mensajes

1

4.f Crear una función que reciba un tipo de trabajo y devuelva un nuevo identificador para dicho trabajo.

```
CREATE FUNCTION fn_NuevoIdentTipoTrabajo(
@elTipoTrabajo varchar(50)
)
RETURNS VARCHAR(150)
AS
BEGIN
    DECLARE @ret VARCHAR(150);
    declare @alphaNumID varchar(10);
    IF(@elTipoTrabajo LIKE 'poster' OR @elTipoTrabajo LIKE 'articulo' OR @elTipoTrabajo LIKE 'capitulo' OR @elTipoTrabajo LIKE '
    BEGIN
        select @alphaNumID = UPPER(SUBSTRING(@elTipoTrabajo, 1, 1));
        declare @ultINS int;
        set @ultINS = (select COUNT(*) from Trabajo where tipoTrab = @elTipoTrabajo);

        set @ret = @alphaNumID + CONVERT(varchar(10), @ultINS);
    END
    ELSE
    BEGIN
        set @ret = 'El parametro ingresado no corresponde a un tipo de trabajo valido.';
    END
    RETURN @ret
END
GO

declare @nueviTipo varchar(150);
set @nueviTipo = dbo.fn_NuevoIdentTipoTrabajo('articulo')
PRINT @nueviTipo
GO
```

83 %

Mensajes

A8

100 %

4.g Crear un procedimiento que dado una universidad devuelva los datos del congreso en el cual público su ultimo Articulo, y algún tipo de indicación de si existe o no un artículo con esas condiciones. Suponemos que no hay congresos que traten los mismos temas y que se solapen en el tiempo.

```
CREATE FUNCTION fn_CongresoDeUniArt(
@laUniversidad varchar(100)
)
RETURNS VARCHAR(300)
AS
BEGIN
declare @nombre VARCHAR(250);
declare @nivellugar int;
declare @diaIni int;
declare @mes int;
declare @anio int;
DECLARE @ret VARCHAR(300);

SELECT @nombre = l.nombre, @nivellugar = l.nivellugar, @diaIni = l.diaIni, @mes = l.mes, @anio = l.año
FROM Lugares l, Trabajo t
WHERE l.idLugar = t.lugarPublic
AND t.idTrab IN (
SELECT MAX(idTrab)
FROM Trabajo ta
WHERE ta.tipoTrab = 'Articulo'
AND lugarPublic IN(
SELECT idLugar
FROM Lugares lg
WHERE lg.tipoLugar = 'Congresos'
AND lg.universidad = @laUniversidad
)
)
set @ret = @nombre+ ' - Nivel Lugar:' + CONVERT(varchar(1), @nivellugar) + ' - Fecha: ' + CONVERT(varchar(2), @diaIni) + '/' + CONVERT(varchar(2), @mes) + '/' + CONVERT(varchar(4), @anio)
RETURN @ret
END
GO
```

83 %

Mensajes

Comandos completados correctamente.

100 %

```
declare @uni varchar(300);
set @uni = dbo.fn_CongresoDeUniArt('UCUDAL')
PRINT @uni
```

83 %

Mensajes

Holiday Inn - Nivel Lugar:2 - Fecha: 10/2/2017

100 %

4.h Crear un procedimiento o función, según considere/corresponda, que dado un tipo de trabajo (Artículo, Poster, etc.) devuelva la cantidad de investigadores que tengan más de 5 trabajos del tipo indicado, y que tengan la maxima cantidad de publicaciones

```
CREATE FUNCTION fn_CantAutTipoTrab( @tipoTrabajo varchar(20))
RETURNS int
AS
BEGIN
    declare @ret int;
    SELECT @ret = COUNT(DISTINCT idInvestigador)
    FROM TAutores ta
    WHERE ta.idInvestigador IN(
        SELECT inv.idInvestigador
        FROM Investigador inv
        WHERE inv.cantTrabPub IN(
            SELECT MAX(cantTrabPub)
            FROM Investigador
        )
    )
    AND ta.idInvestigador IN(
        SELECT inv.idInvestigador
        FROM Investigador inv, Trabajo t, TAutores ta
        WHERE inv.idInvestigador = ta.idInvestigador
        AND ta.idTrab = t.idTrab
        AND t.tipoTrab LIKE @tipoTrabajo
        GROUP BY inv.idInvestigador
        HAVING COUNT(*)>5
    )

    RETURN @ret;
END
go

declare @ret int;
set @ret = dbo.fn_CantAutTipoTrab('articulo')
PRINT @ret
```

83 %

Mensajes

2

100 %

## Script con la creación de disparadores

5 - a. Crear un trigger que al insertar un trabajo asegure las restricciones identificadas, y en caso de que no se asignó una fecha de inicio el trigger asigne el primero de enero del año actual como fecha de inicio del trabajo.

En el caso de considerarse insert de a un trabajo por vez, debe asegurarse de que la operación se haga solo en ese caso.

```
CREATE TRIGGER tg_RestriccionesTrabajo
ON Trabajo
INSTEAD OF INSERT
AS
BEGIN
    IF( NOT EXISTS( Select COUNT(*) From inserted GROUP BY nomTrab having count(*) > 1))
    BEGIN
        DECLARE @nomTrab VARCHAR(100);
        DECLARE @descripTrab VARCHAR(100);
        DECLARE @tipoTrab VARCHAR(20);
        DECLARE @fecha date;

        /*Trigger Trabajo alfanumerico*/
        declare @ultINS int;
        set @ultINS = (select COUNT(*) from Trabajo where tipoTrab in (select tipoTrab from inserted));
        declare @alphaNumID varchar(10);
        select @alphaNumID = UPPER(SUBSTRING(tipoTrab, 1, 1)) from inserted;
        set @alphaNumID = @alphaNumID + CONVERT(varchar(10), @ultINS);

        /* Trigger Sa */
        select @nomTrab = nomTrab, @descripTrab = descripTrab, @tipoTrab = tipoTrab, @fecha = fechaInicio
        from inserted

        IF(@nomTrab is not null AND @tipoTrab is not null)
        BEGIN
            IF(@tipoTrab like 'poster' OR @tipoTrab like 'articulo' OR @tipoTrab like 'capitulo' OR @tipoTrab like 'otro')
            BEGIN
                if(@fecha is null) Begin set @fecha = '01-01-'+CONVERT(varchar(4),YEAR(GETDATE())); End
                insert into Trabajo
                select nomTrab, descripTrab, tipoTrab, @fecha, linkTrab, lugarPublic, @alphaNumID
                from inserted

            END
            ELSE
            BEGIN
                PRINT 'Parametros no correctos'
            END
        END

        END
        ELSE
        BEGIN
            PRINT 'No se admiten inserciones múltiples.'
        END
    END
GO
```

83 %

Mensajes

Comandos completados correctamente.

100 %



INSERT INTO Trabajo  
VALUES ('Investigacion sobre Fecha Inicio',  
'Investigacionque estudia como no poner una Fecha Inicio',  
'articulo', null, 'www.google.com',4, 'id')

83 %

Mensajes

(1 filas afectadas)

(1 filas afectadas)

100 %

select \* from Trabajo

83 %

Resultados Mensajes

	nomTrab	descripTrab	tipoTrab	fechaInicio	linkTrab
9	Investigacion sobre Fecha l...	Investigacionque estudia como no poner una Fecha...	articulo	2017-01-01	www.google.com
10	Venado de campo	Investigacion sobre el venado de campo, uno de los...	capitulo	2017-02-01	http://blogs.ceibal.edu.uy/formacion/
11	Investigacion sobre Bases d...	Una base de datos o banco de datos es un conjunt...	capitulo	2017-12-08	https://es.wikipedia.org/wiki/Base_de

5b - Crear un trigger que controle que solo puedan eliminarse trabajos no publicados que iniciaron hace más de 2 años. Eliminando todos los datos de la base de datos que considere necesarios asociados a dichos trabajos.  
Debe considerarse eliminaciones múltiples

```
CREATE TRIGGER tg_EliminarTrabajos
ON Trabajo
INSTEAD OF DELETE
AS
BEGIN
    DELETE FROM Referencias
    WHERE idTrab IN (
        SELECT x.idTrab
        FROM Trabajo x
        WHERE YEAR(x.fechaInicio) < (YEAR(GETDATE()) - 2)
    )
    AND idTrab IN (
        SELECT y.idTrab
        FROM Trabajo y
        WHERE y.lugarPublic IS NULL
    )
    AND idTrab IN(
        SELECT idTrab
        FROM deleted
    )

    DELETE FROM Referencias
    WHERE idTrabReferenciado IN (
        SELECT x.idTrab
        FROM Trabajo x
        WHERE YEAR(x.fechaInicio) < (YEAR(GETDATE()) - 2)
    )
    AND idTrabReferenciado IN (
        SELECT y.idTrab
        FROM Trabajo y
        WHERE y.lugarPublic IS NULL
    )
)
```

```
AND idTrabReferenciado IN(
    SELECT idTrab
    FROM deleted
)
)
DELETE FROM TAutores
WHERE idTrab IN (
    SELECT x.idTrab
    FROM Trabajo x
    WHERE YEAR(x.fechaInicio) < (YEAR(GETDATE()) - 2)
)
AND idTrab IN (
    SELECT y.idTrab
    FROM Trabajo y
    WHERE y.lugarPublic IS NULL
)
AND idTrab IN(
    SELECT idTrab
    FROM deleted
)
)
DELETE FROM TTags
WHERE idTrab IN (
    SELECT x.idTrab
    FROM Trabajo x
    WHERE YEAR(x.fechaInicio) < (YEAR(GETDATE()) - 2)
)
AND idTrab IN (
    SELECT y.idTrab
    FROM Trabajo y
    WHERE y.lugarPublic IS NULL
)
)
)
)
AND idTrab IN(
    SELECT idTrab
    FROM deleted
)
)
DELETE FROM Trabajo
WHERE YEAR(fechaInicio) < (YEAR(GETDATE()) - 2)
AND lugarPublic IS NULL
AND idTrab IN(
    SELECT idTrab
    FROM deleted
)
)
END
GO
```

83 %

Mensajes

Comandos completados correctamente.

100 %

```
INSERT INTO Trabajo
VALUES ('Investigacion de año actual No publicada',
'Investigacion que estudia como NO eliminar un trabajo de este año, no publicada',
'articulo', '02-02-2017', 'www.google.com',null, 'id')
Delete from Trabajo where idTrab='A12'
```

83 %

Mensajes

(0 filas afectadas)

(0 filas afectadas)

(0 filas afectadas)

(0 filas afectadas)

(0 filas afectadas)

(1 filas afectadas)

```
INSERT INTO Trabajo
VALUES ('Investigacion de hace 3 años No publicada',
'Investigacion que estudia como SI eliminar un trabajo de hace 3 años, no publicada',
'articulo', '02-02-2014', 'www.google.com',null, 'id')
Delete from Trabajo where idTrab='A13'
```

83 %

Mensajes

(0 filas afectadas)

(0 filas afectadas)

(0 filas afectadas)

(0 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

100 %

5c - Crear un trigger que solo permita insertar dos trabajos, uno como referencia del otro, si tienen algún tema (palabras claves) en común.

```
CREATE TRIGGER tg_InsertarReferencias
ON Referencias
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @trabajo VARCHAR(10)

    SELECT @trabajo = x.idTrab
    FROM inserted x
    WHERE idTrab IN(
        SELECT idTrab
        FROM TTags
        WHERE idTag IN
        (
            SELECT idTag
            FROM TTags
            WHERE idTrab IN(
                SELECT y.idTrabReferenciado
                FROM inserted y
            )
        )
    )

    IF (@trabajo IS NOT NULL)
    BEGIN
        INSERT INTO Referencias
        SELECT idtrab, idTrabReferenciado
        FROM inserted
    END
    ELSE
    BEGIN
        PRINT 'Los trabajos no tienen temas (palabras clave) en común.'
    END
END
GO
```

75 %

Mensajes

Comandos completados correctamente.

100 %

select \* from TTags

91 % <

Resultados Mensajes

	idTrab	idTag
1	A0	1
2	A0	3
3	C0	5
4	O0	7
5	O1	9
6	P0	15
7	O1	27
8	A2	29
9	A3	29
10	A4	29
11	C1	29
12	A2	31
13	A3	33
14	A4	35
15	A5	37
16	A6	39
17	A5	41
18	A6	41
19	A7	41

INSERT INTO Referencias  
VALUES('A0', 'C0')

91 % <

Mensajes

Los trabajos no tienen temas (palabras clave) en común.  
(1 filas afectadas)

INSERT INTO Referencias  
VALUES('A2', 'A3')

91 % <

Mensajes

(1 filas afectadas)  
(1 filas afectadas)

5d - Crear un trigger que audite las inserciones y las actualizaciones de los lugares que tienen algún trabajo publicado, registrando en alguna tabla auxiliar el lugar que se quiere actualizar/insertar, la operación (INS o UPD), el usuario y la fecha hora de la operación.

```
CREATE TABLE LogInsertAndUpdate (  
    lugar INT,  
    operacion VARCHAR(3) CHECK (operacion IN ('INS', 'UPD')),  
    usuario VARCHAR(50),  
    fechaOperacion date  
)  
GO
```

91 %

Mensajes

Comandos completados correctamente.

```
CREATE TRIGGER tg_AuditarInsOrUpd  
ON Lugares  
AFTER INSERT, UPDATE  
AS  
BEGIN  
    DECLARE @lugar INT  
    SELECT @lugar = idLugar  
    FROM inserted  
  
    IF(EXISTS (SELECT * FROM inserted) AND NOT EXISTS (SELECT * FROM deleted))  
    BEGIN  
        INSERT INTO LogInsertAndUpdate  
        VALUES (@lugar, 'INS', USER, GETDATE())  
    END  
    ELSE IF(EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted))  
    BEGIN  
        IF(EXISTS (  
            SELECT *  
            FROM inserted  
            WHERE idLugar IN  
            (  
                SELECT lugarPublic  
                FROM Trabajo  
            )  
        )  
        )  
        BEGIN  
            INSERT INTO LogInsertAndUpdate  
            VALUES (@lugar, 'UPD', USER, GETDATE())  
        END  
    END  
END
```

83 %

Mensajes

Comandos completados correctamente.

```

INSERT INTO Lugares
VALUES
(6, 'Test registro 1', 4, 2016, 11, 8, null, 'http://www.teatrosolis.org.uy', 'Udelar', 'Revistas')

INSERT INTO Lugares
VALUES
(7, 'Test registro 2', 4, 2016, 11, 8, null, 'http://www.teatrosolis.org.uy', 'Udelar', 'Revistas')

Update Lugares set link='www.google.com' where idLugar=7 /*No tiene lugar publicado, no debe aparecer*/
Update Lugares set link='www.latu.ed.uy' where idLugar=2 /*Si tiene lugar publicado, debe aparecer*/

select * from LogInsertAndUpdate

```

83 %

Resultados Mensajes

	lugar	operacion	usuario	fechaOperacion
1	6	INS	dbo	2017-12-11
2	7	INS	dbo	2017-12-11
3	2	UPD	dbo	2017-12-11

## Script con la resolución de las consultas y vista

6 - a. Mostrar, para cada trabajo publicado con mas de 3 autores, el identificador del trabajo, nombre del mismo, la cantidad de autores que tiene, y lugar donde se publico (id y nombre)

```

/*
a. Mostrar, para cada trabajo publicado con mas de 3 autores, |
el identificador del trabajo, nombre del mismo, la cantidad de autores que tiene,
y lugar donde se publico (id y nombre).
*/

SELECT t.idTrab as 'Identificador del trabajo', t.nomTrab as 'Nombre de Trabajo',
Count(distinct ta.idInvestigador) as 'Cantidad de Autores', l.idLugar as 'Id Lugar', l.nombre as 'Nombre del Lugar'
FROM Trabajo t, Lugares l, TAutores ta
WHERE t.lugarPublic = l.idLugar and t.idTrab = ta.idTrab
GROUP BY t.idTrab, t.nomTrab, l.idLugar, l.nombre
HAVING COUNT(*) > 3

```

83 %

Resultados Mensajes

	Identificador del trabajo	Nombre de Trabajo	Cantidad de Autores	Id Lugar	Nombre del Lugar
1	A1	Investigacion sobre las drogas	4	4	Holiday Inn

6 - b. Obtener la lista de palabras claves que aparecen en trabajos sobre 'BASE DE DATOS' (palabra clave) publicados en revistas del año actual. La salida debe aparecer ordenado alfabéticamente, sin repetidos

```
/*
b. Obtener la lista de palabras claves que aparecen en trabajos sobre 'BASE DE DATOS'
(palabra clave) publicados en revistas del año actual. La salida debe aparecer
ordenado alfabéticamente, sin repetidos.
*/

SELECT DISTINCT t.palabra
FROM TTags tt, Tags t, Trabajo w, Lugares l
WHERE tt.idTag = t.idtag
AND tt.idTrab = w.idTrab
AND w.lugarPublic = l.idLugar
AND l.tipoLugar LIKE 'Revistas'
AND l.año = YEAR(GETDATE())
AND w.idTrab IN(
    SELECT z.idTrab
    FROM TTags z
    WHERE z.idTag IN (
        SELECT y.idtag
        FROM Tags y
        WHERE y.palabra = 'BASE DE DATOS'
    )
)
ORDER BY (t.palabra)
GO
```

91 %

Resultados Mensajes

	palabra
1	BASE DE DATOS
2	oracle
3	sql
4	xquery



6 - c - Mostrar los datos de las universidades y el link a los congresos, de las universidades que han realizado más de 2 congresos de nivel 4, en los ultimos 5 años, con mas de 20 trabajos publicados en esos congresos.

```

/* c - Mostrar los datos de las universidades y el link a los congresos,
de las universidades que han realizado más de 2 congresos de nivel 4,
en los ultimos 5 años, con mas de 20 trabajos publicados en esos congresos. */

SELECT DISTINCT u.nombre, l.link
FROM Universidad u, Lugares l
WHERE u.nombre = l.universidad
AND l.universidad IN (
    SELECT x.universidad
    FROM Lugares x
    WHERE x.tipoLugar LIKE 'Congresos'
    GROUP BY x.universidad
    HAVING COUNT (*) > 0 /* Se cambió a 0 para mostrar resultados
                        consulta original: HAVING COUNT (*) > 20 */
)
AND l.nivelLugar = 4
AND l.año > YEAR(GETDATE())-5
AND l.tipoLugar LIKE 'Congresos'
GROUP BY u.nombre, l.link
HAVING COUNT(*) > 0 /* Se cambió a 0 para mostrar resultados
                    consulta original: HAVING COUNT (*) > 2 */
GO

```

	nombre	link
1	ORT	www.latu.ed.uy

6 - d- Obtener para cada investigador el ultimo trabajo que inicio en el cual fue/es autor principal.

```

/* d- Obtener para cada investigador el ultimo trabajo
que inicio en el cual fue/es autor principal.*/

SELECT i.idInvestigador, ta.idTrab
FROM Investigador i LEFT OUTER JOIN TAutores ta
ON i.idInvestigador = ta.idInvestigador
LEFT OUTER JOIN Trabajo t
ON ta.idTrab = t.idTrab
WHERE ta.rolinvestig = 'autor-ppal'
AND t.idTrab IN
(
    SELECT TOP 1 x.idTrab
    FROM Trabajo x, TAutores y
    WHERE x.idTrab = y.idTrab
    AND i.idInvestigador = y.idInvestigador
    ORDER BY fechaInicio DESC
)
GO

```

	idInvestigador	idTrab
1	1	A0
2	4	A1
3	6	A1
4	11	A2
5	12	A2
6	2	O1



6 - e - Para cada investigador mostrar, su identificación, nombre, nombre de la universidad a la que pertenece, y la cantidad de trabajos suyos publicados en lugares de nivel 1, de nivel 2, de nivel 3 y de nivel 4, en los últimos 5 años, en la carrera de Ingeniería.

```

/* e - Para cada investigador mostrar, su identificación,
nombre, nombre de la universidad a la que pertenece,
y la cantidad de trabajos suyos publicados en lugares de nivel 1,
de nivel 2, de nivel 3 y de nivel 4,
en los últimos 5 años, en la carrera de Ingeniería. */
CREATE FUNCTION fn_CantTrabajoPorNivel(
@nivelLugar int,
@idInv int
)
RETURNS int
AS
BEGIN
    DECLARE @ret int
    SELECT @ret = COUNT(*)
    FROM Lugares lu, Investigador inv, TAutores x, Trabajo t
    WHERE inv.idInvestigador = x.idInvestigador
    AND lu.nivelLugar = @nivelLugar
    AND x.idTrab = t.idTrab
    AND inv.idInvestigador = @idInv
    AND t.lugarPublic = lu.idLugar
    AND YEAR(t.fechaInicio) > YEAR(GETDATE())-5
    AND inv.carrera LIKE 'Ingeniería'
    RETURN @ret
END
GO

```

Mensajes

Comandos completados correctamente.

```

SELECT DISTINCT i.idInvestigador, i.nombre, i.idUniversidad,
dbo.fn_CantTrabajoPorNivel(1, i.idInvestigador) as 'Cantidad trabajos nivel 1',
dbo.fn_CantTrabajoPorNivel(2, i.idInvestigador) as 'Cantidad trabajos nivel 2',
dbo.fn_CantTrabajoPorNivel(3, i.idInvestigador) as 'Cantidad trabajos nivel 3',
dbo.fn_CantTrabajoPorNivel(4, i.idInvestigador) as 'Cantidad trabajos nivel 4'
FROM Investigador i LEFT OUTER JOIN TAutores ta
ON i.idInvestigador = ta.idInvestigador
LEFT OUTER JOIN Trabajo t
ON ta.idTrab = t.idTrab
LEFT OUTER JOIN Lugares l
ON t.lugarPublic = l.idLugar
GO

```

Resultados

	idInvestigador	nombre	idUniversidad	Cantidad trabajos nivel 1	Cantidad trabajos nivel 2	Cantidad trabajos nivel 3	Cantidad trabajos nivel 4
1	1	Marcelo López	Udelar	0	0	0	0
2	2	Laura Marquisio	UM	0	0	0	0
3	3	Fabián Méndez	Udelar	0	0	0	0
4	4	Silvia Luque	Udelar	0	0	0	0
5	5	Silvio Duarte	Udelar	0	0	0	0
6	6	Walter Clitish	Udelar	0	0	0	0
7	7	Maicol Uriarte	UBA	0	0	0	0
8	8	Mariela Ifrán	Universidad de Córdoba	0	0	0	0
9	9	Linda Cibils	Universidad de Amazonas	0	0	0	0
10	10	Marcio Avellanah	Universidad Federal de Alagoas	0	0	0	0
11	11	Guillermo Polachek	ORT	4	3	0	0
12	12	Sebastian Villar	Universidad de Amazonas	4	3	0	0
13	13	Atuagualpo Galpones	ORT	1	1	0	0

6 - f - Para la universidad "ORT" mostrar el identificador de sus investigadores que tienen algún trabajo en el año actual y la cantidad de trabajos publicados en congresos de nivel 4.

En caso de investigadores sin trabajos publicados en estos congresos pero con trabajos en proceso el año actual deben aparecer en el resultado.

```

/* f - Para la universidad "ORT" mostrar el identificador de sus investigadores
que tienen algún trabajo en el año actual y la cantidad de trabajos
publicados en congresos de nivel 4.
En caso de investigadores sin trabajos publicados en estos congresos pero con
trabajos en proceso el año actual deben aparecer en el resultado.*/

SELECT i.idInvestigador, (
    SELECT COUNT(DISTINCT tra.idTrab)
    FROM Lugares lu, TAutores x, Trabajo tra
    WHERE lu.idLugar = tra.lugarPublic
    AND i.idInvestigador = x.idInvestigador
    AND tra.idTrab = x.idTrab
    AND lu.nivelLugar = 4
    AND lu.tipoLugar = 'Congresos'
) AS 'Cant. de Trabajos en ORT para Nivel 4'
FROM Investigador i LEFT OUTER JOIN TAutores ta
ON i.idInvestigador = ta.idInvestigador
LEFT OUTER JOIN Trabajo t
ON ta.idTrab = t.idTrab
WHERE i.idUniversidad LIKE 'ORT'
AND YEAR(t.fechaInicio) = YEAR(GETDATE())
GROUP BY i.idInvestigador

```

100 %

Resultados Mensajes

	idInvestigador	Cant. de Trabajos en ORT para Nivel 4
1	11	0
2	13	0

6 - g - Mostrar para cada universidad que tiene trabajos publicados, los datos del último trabajo publicado. Solucionar usando la función a).

```

/* g - Mostrar para cada universidad que tiene trabajos publicados,
los datos del último trabajo publicado.
Solucionar usando la función a).*/

SELECT DISTINCT U.nombre, dbo.fn_UltimoTrabajoPorUniv(l.universidad) AS 'Datos de Trabajo publicado'
FROM Trabajo t, Lugares l, Universidad u
WHERE t.lugarPublic = l.idLugar
AND U.nombre = L.universidad

```

100 %

Resultados Mensajes

	nombre	Datos de Trabajo publicado
1	ORT	C0 - Venado de campo
2	UBA	C1 - Investigacion sobre Bases de Datos
3	UCUDAL	A5 - Investigacion sobre Windows
4	Udelar	A0 - Investigacion GARZA CUCA
5	UM	P0 - Investigacion sobre el Agua

6 - h- Eliminar las palabras claves no usadas en los trabajos.

```

DELETE FROM Tags
WHERE idTag NOT IN
(
    SELECT idtag
    FROM TTags
)
GO

```

Mensajes

(11 filas afectadas)

select \* from Tags

100 %

Resultados Mensajes

	palabra	idTag
1	garza	1
2	cuca	3
3	venado	5
4	campo	7
5	fauna	9
6	animales	11
7	silvestre	13
8	agua	15
9	ambiente	17
10	ecología	19
11	drogas	21
12	adicciones	23
13	cultura	25
14	mayas	27
15	BASE D...	29
16	sql	31
17	xquery	33
18	oracle	35
19	windows	37
20	ubuntu	39
21	sistema-o...	41
22	Colon	43
23	America	45
24	Program...	47

select \* from Tags

100 %

Resultados Mensajes

	palabra	idTag
1	garza	1
2	cuca	3
3	venado	5
4	campo	7
5	fauna	9
6	agua	15
7	mayas	27
8	BASE DE DATOS	29
9	sql	31
10	xquery	33
11	oracle	35
12	windows	37
13	ubuntu	39
14	sistema-operativo	41

7 - a- Realizar una vista que muestre lista de Congresos y para cada uno la cantidad de trabajos publicados que no tienen autores que sean investigadores de la universidad anfitriona del congreso

```
/* a- Realizar una vista que muestre lista de Congresos y
para cada uno la cantidad de trabajos publicados que no tienen autores que
sean investigadores de la universidad anfitriona del congreso.
*/

create VIEW View_ListaCongresos
AS
SELECT lu.nombre 'Congreso', COUNT(DISTINCT tra.idTrab) AS 'Cantidad de trabajos'
FROM Lugares lu, Investigador inv, TAutores x, Trabajo tra
WHERE lu.idLugar = tra.lugarPublic
AND inv.idInvestigador = x.idInvestigador
AND tra.idTrab = x.idTrab
AND lu.tipoLugar LIKE 'Congresos'
AND tra.idTrab NOT IN (
    SELECT t.idTrab
    FROM Investigador i, Trabajo t, TAutores ta, Lugares l
    WHERE t.idTrab = ta.idTrab
    AND i.idInvestigador = ta.idInvestigador
    AND i.idUniversidad = lu.Universidad
)
GROUP BY lu.nombre
GO
```

100 %

Mensajes

Comandos completados correctamente.

100 %

Resultados Mensajes

	Congreso	Cantidad de trabajos
1	Holiday Inn	4
2	LATU	1

7 - b- Realizar una vista que muestre para cada Investigador, para cada tipo de trabajo la fecha de inicio del primer y último trabajo.  
 Todas los investigadores deben aparecer en el resultado, aunque no tengan trabajos que cumplan las condiciones

```

/* b- Realizar una vista que muestre para cada Investigador,
para cada tipo de trabajo la fecha de inicio del primer y último trabajo.
Todas los investigadores deben aparecer en el resultado,
aunque no tengan trabajos que cumplan las condiciones.
*/
CREATE VIEW View_TrabajoInvestigadores
AS
SELECT i.idInvestigador Investigador, t.tipoTrab 'Tipo de trabajo',
      MIN(t.fechaInicio) as 'Fecha Inicio primer Trabajo',
      MAX(t.fechaInicio) as 'Fecha Inicio ultimo Trabajo'
FROM Investigador i LEFT OUTER JOIN TAutores ta
ON i.idInvestigador = ta.idInvestigador
LEFT OUTER JOIN Trabajo t
ON ta.idTrab = t.idTrab
GROUP BY i.idInvestigador, t.tipoTrab
GO
  
```

Mensajes  
Comandos completados correctamente.

`SELECT * FROM View_TrabajoInvestigadores`

	Investigador	Tipo de trabajo	Fecha Inicio primer Trabajo	Fecha Inicio ultimo Trabajo
1	7	NULL	NULL	NULL
2	10	NULL	NULL	NULL
3	1	articulo	2016-04-03	2016-04-03
4	4	articulo	2017-05-17	2017-05-17
5	5	articulo	2016-04-03	2016-04-03
6	6	articulo	2017-05-17	2017-05-17
7	8	articulo	2017-05-17	2017-05-17
8	9	articulo	2016-04-03	2017-05-17
9	11	articulo	2017-11-08	2017-12-08
10	12	articulo	2017-11-08	2017-12-08
11	13	articulo	2017-11-09	2017-11-09
12	2	capitulo	2017-02-01	2017-02-01
13	11	capitulo	2017-12-08	2017-12-08
14	12	capitulo	2017-12-08	2017-12-08
15	13	capitulo	2017-12-08	2017-12-08
16	2	Otro	2017-04-28	2017-04-28
17	3	poster	2017-05-17	2017-05-17

## **BASES DE DATOS 2**

Obligatorio

Estudiantes: Guillermo Polachek (153924) - Sebastian Villar (177751)

Diciembre 2017