INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

upiita-ipr

INGENIERÍA TELEMATICA

UNIDAD DE APRENDIZAJE: BASE DE DATOS DISTRIBUIDAS PRIMER EXAMEN DEPARTAMENTAL

NOMBRE: _____Sanchez Ramirez Noel Adan_____. GRUPO: __3TV2___

Calificación: _____ FECHA: 01 de diciembre de 2020

INSTRUCCIONES: RESOLVER TODOS LOS REACTIVOS Y PROBLEMAS.
Escribir de forma clara y concisa (indicar procedimiento y resultado).
Puntuación total del examen 10pts (el examen consta de 4 páginas).

FAVOR DE APAGAR EL CELULAR O COMPUTADORA, **DE NO ATENDER LA INDICACIÓN SE LE CANCELARA EL EXAMEN Y SU CALIFICACIÓN SERA DE CERO.**

1. ¿Qué es una base de datos? **Es una colección de datos relacionados, los cuáles tienen un significado implícito.**

2. ¿Qué es una base de datos distribuidas? **Es la forma en cómo la colección o ciertas colecciones de datos deben de estar localizados.**

3. ¿Qué es normalización? **Es un proceso en el cuál organizamos y designamos de forma eficiente los datos, para eliminar redundancia y asegurar dependencias con sentido.**

4. Describa cada una de las formas normales, Explique con un ejemplo.

1NF-> Definir los campos que usaremos, asegurar que no haya datos repetidos y que al menos exista una llave primaria por tabla.

2NF-> Asegurar que no haya dependencias parciales en las columnas de las llaves primarias.

3NF-> Asegurar que los datos tengan dependencia hacia la llave primaria y que la tabla al menos este en 2NF.

5. ¿Qué es una restricción? **Son reglas que definen o establecen a los datos de las columnas, sirven para limitar los datos que serán almacenados.**

6. Escriba 5 ejemplos de restricciones

NOT NULL, DEFAULT, PRIMARY KEY, UNIQUE Y CHECK

7. Realice un Store Procedure que agregue una ‘x’ columna a una ‘y’ tabla, indicando como parámetros: nombre de tabla, nombre de columna, tipo de dato de la columna, tamaño de la columna (si aplica), si acepta Nulos o no, valor por default en caso de no aceptar nulos.

La ejecución será de la siguiente forma:

```
exec AddColumn_Tabla 'T_PRUEBA','Fecha','DATETIME',NULL,1,'GETDATE()'
```

La salida deberá ser:

1) La tabla con la nueva columna

	ID	Nombre	Fecha
1	1	dato 1	2020-11-30 21:48:06.210
2	2	dato 2	2020-11-30 21:48:06.210

*Se deben considerar todas las validaciones necesarias, en caso de ingresar algún dato erróneo se deberá avisar que dato fue incorrecto.

```
CREATE OR ALTER PROCEDURE SP_TABLAS @NOMBRE NVARCHAR(25), @COLUMNA NVARCHAR(25),
@TIPO NVARCHAR(20), @Null NVARCHAR(10), @TAM NVARCHAR(25), @DEFAULT NVARCHAR(20)
AS
BEGIN
	DECLARE @QUERY AS NVARCHAR(500);
	SET @QUERY = ' ALTER TABLE ' + @NOMBRE + ' ADD ' + @COLUMNA + ' ' + @TIPO + '
' + @Null + ' DEFAULT ' + @DEFAULT;
	PRINT @QUERY;
	BEGIN TRY
		EXEC(@QUERY);
	END TRY
	BEGIN CATCH
		SELECT ERROR_MESSAGE();
	END CATCH

END
```


INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN
INGENIERÍA Y TECNOLOGÍAS AVANZADAS
upiita-ipr

INGENIERÍA TELEMATICA
UNIDAD DE APRENDIZAJE: BASE DE DATOS
DISTRIBUIDAS PRIMER EXAMEN DEPARTAMENTAL

2.- Con base en la siguiente información:

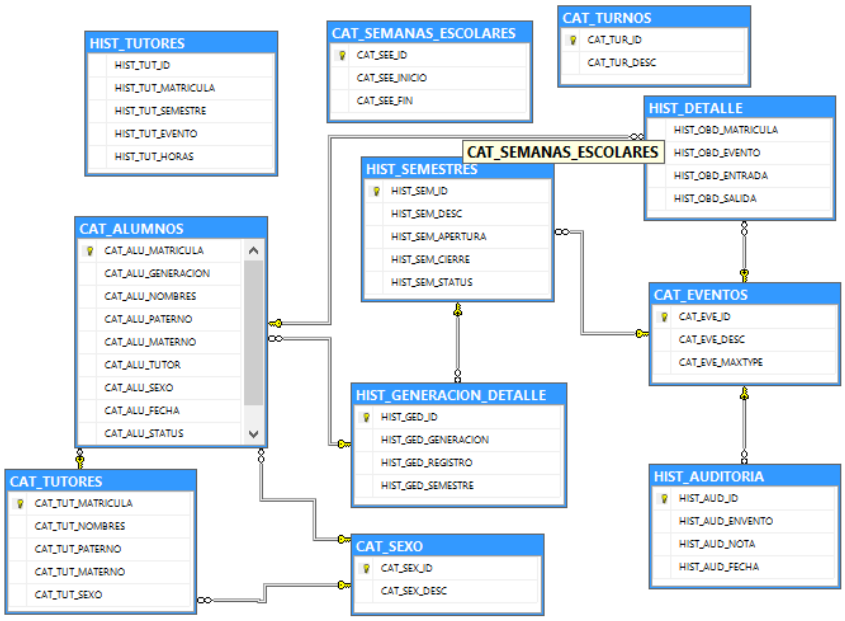
Alumnos			Matricula	Tutor	Observacion (150)	Simulacion (360)	Familiarizacion (240)						Especializacion (480)				TOTAL
							Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Total	Sem 1	Sem	Sem	Total	
Rosas	Mejia	Alma	A182021292	Karina	150	360	180	70				250	480			480	1240
Aparicio	Nava	Julio Cesar	A182020261	Alejandra	150	180			150		150	300				0	630
De La Rosa	Mendoza	Javier	A172018552	Saúl	150	360		240				240	480			480	1230
Del Carmen	Briseño	Ricardo	B172029962	Leticia	150	360	20					20	120	240	480	840	1010
Fernandez	Garcia	Jose Daniel	A162018559	Alejandra	150	360	60	20	40	18	110	248	240	240		480	1230
Fernandez	Gonzalez	Samuel	A202019071	Saúl	150	360	60	8	36	136		240	240	240		480	1230
Feria	Guerreros	Itzel	B162021689	Karina	150	360						0	240			240	750
Flores	Lopez	Mauricio	B192019195	Leticia	150	360	60	60	50			170	480			480	1160
Franco	Agueda	Gustavo	A182018838	Alejandra	150	360						0	720			720	990
Lozano	Espinoza	Jose Daniel	A152021574	Leticia	150	360	240					240	240	240		480	1230
Maldonado	Rodriguez	Samuel	A182020124	Saúl	150	360	60	200				260	480			480	1250
Mayen	Urrua	Itzel	B182019886	Karina	10	360	240					240	480			480	1090
Mendoza	Arias	Mauricio	A202018675	Leticia	150	360	240					240	480			480	1230
Muñoz	Martinez	Gustavo	B172019776	Alejandra	150	360	120	190				310	480			480	1210

1) Se requiere una base que almacene la cantidad de horas por Tipo de práctica (Observación, Simulación, Familiarización y Especialización) de cada alumno. Teniendo en

cuenta las siguientes consideraciones

a) Las prácticas de Familiarización y Especialización se van registrando por semestre.

2) La base creada debe estar Normalizada



3) Generar una consulta que muestre el total de horas por alumno y por tipo de práctica (sin considerar el semestre) tomar en cuenta que el tope de horas por tipo de práctica es el que se muestra en el paréntesis, aunque el alumno haya registrado más horas de esa cantidad el excedente no deberá ser contemplado en el Total

```
	WITH DETALLE AS(
		SELECT HIST_OBD_MATRICULA MATRICULA, HIST_OBD_EVENTO EVENTO,
SUM((DATEDIFF(HOUR,HIST_OBD_ENTRADA,HIST_OBD_SALIDA))) HORAS
FROM HIST_DETALLE
GROUP BY HIST_OBD_MATRICULA,HIST_OBD_EVENTO
	)
	SELECT D.MATRICULA,
	D.EVENTO,
	IIF((SELECT CE.CAT_EVE_MAXTYPE FROM CAT_EVENTOS CE WHERE CE.CAT_EVE_ID =
D.EVENTO) < SUM(D.HORAS), (SELECT CE.CAT_EVE_MAXTYPE FROM CAT_EVENTOS CE WHERE
CE.CAT_EVE_ID = D.EVENTO), SUM(D.HORAS))
	"TOTAL HORAS" FROM DETALLE D
	GROUP BY D.MATRICULA, D.EVENTO, D.HORAS;
```

4) Generar un programa (T-SQL) que reciba como parámetro la matrícula del alumno y regrese el excedente de horas que tenga por tipo de práctica

```
CREATE OR ALTER PROCEDURE SP_EXCEDENTE @MATRICULA NVARCHAR(20)
AS
BEGIN
	DECLARE @ID NVARCHAR(20) = SUBSTRING(@MATRICULA,7,LEN(@MATRICULA));
	WITH DETALLE AS(
		SELECT HIST_OBD_MATRICULA MATRICULA, HIST_OBD_EVENTO EVENTO,
SUM((DATEDIFF(HOUR,HIST_OBD_ENTRADA,HIST_OBD_SALIDA))) HORAS
FROM HIST_DETALLE WHERE HIST_DETALLE.HIST_OBD_MATRICULA = CAST(@ID AS INTEGER)
GROUP BY HIST_OBD_MATRICULA,HIST_OBD_EVENTO
```

```
        )
        SELECT D.MATRICULA,
        D.EVENTO,
        (SUM(HORAS) - (SELECT CE.CAT_EVE_MAXTYPE FROM CAT_EVENTOS CE WHERE
CE.CAT_EVE_ID = D.EVENTO))
        "HORAS EXCEDENTES" FROM DETALLE D
        GROUP BY D.MATRICULA, D.EVENTO, D.HORAS;
END

    EXEC SP_EXCEDENTE 'A202010001';
```

5) Generar un programa (T-SQL) que reciba como parámetro el tipo de Práctica y regrese la lista de alumnos con sus respectivos “totales” de horas en dicho tipo de práctica (separar total sin excedente y el excedente registrado)

```
CREATE OR ALTER PROCEDURE SP_HORAS @TIPO NVARCHAR(20)
AS
BEGIN
        WITH DETALLE AS(
                SELECT HIST_OBD_MATRICULA MATRICULA, HIST_OBD_EVENTO EVENTO,
SUM((DATEDIFF(HOUR,HIST_OBD_ENTRADA,HIST_OBD_SALIDA))) HORAS
FROM HIST_DETALLE WHERE HIST_OBD_EVENTO = @TIPO
GROUP BY HIST_OBD_MATRICULA,HIST_OBD_EVENTO
        )
        SELECT D.MATRICULA,
        D.EVENTO,
        HORAS “REGISTRADO”,
        IIF((SELECT CE.CAT_EVE_MAXTYPE FROM CAT_EVENTOS CE WHERE CE.CAT_EVE_ID =
D.EVENTO) < SUM(D.HORAS), (SELECT CE.CAT_EVE_MAXTYPE FROM CAT_EVENTOS CE WHERE
CE.CAT_EVE_ID = D.EVENTO), SUM(D.HORAS)) "HORAS",
        (SUM(HORAS) - (SELECT CE.CAT_EVE_MAXTYPE FROM CAT_EVENTOS CE WHERE
CE.CAT_EVE_ID = D.EVENTO))
        "HORAS EXCEDENTES" FROM DETALLE D
        GROUP BY D.MATRICULA, D.EVENTO, D.HORAS;
END

    EXEC SP_HORAS 3;
```

6) Crear un trigger que registre en la tabla del alumno un “Estatus” (Vacío, Incompleto, Completo) el cual debe cambiar cuando se agreguen horas en cualquiera de los tipos de prácticas

a. Si el alumno no tiene ninguna práctica (o acaba de registrarse) debe iniciar en “Vacío”

b. Si se ha registrado por lo menos 1 hora y menos del Total final (1230 horas) debe estar en “Incompleto”

c. Si se llegan a las 1230 horas debe cambiar a “Completo”

d. El total de horas no contempla excedentes por Tipo de práctica

```
CREATE OR ALTER TRIGGER TG_STATUS
ON HIST_DETALLE
FOR UPDATE
AS BEGIN
        DECLARE @FECHAI DATETIME = (SELECT inserted.HIST_OBD_ENTRADA FROM inserted);
        DECLARE @FECHAF DATETIME = (SELECT inserted.HIST_OBD_SALIDA FROM inserted);
        DECLARE @TIPO INTEGER = (SELECT inserted.HIST_OBD_EVENTO FROM inserted);
        DECLARE @MATRICULA INTEGER = (SELECT inserted.HIST_OBD_MATRICULA FROM
inserted);
        DECLARE @HORAS INTEGER = (SELECT
        SUM((DATEDIFF(HOUR,HD.HIST_OBD_ENTRADA,HD.HIST_OBD_SALIDA)))
```

```
	FROM HIST_DETALLE HD WHERE HD.HIST_OBD_MATRICULA = @MATRICULA );
	DECLARE @CONTADOR INTEGER = (SELECT COUNT(*) FROM HIST_DETALLE HD WHERE
HD.HIST_OBD_MATRICULA = @MATRICULA);
	IF(@CONTADOR = 0)
		BEGIN
			UPDATE CAT_ALUMNOS SET CAT_ALU_STATUS = 0 WHERE
CAT_ALU_MATRICULA = @MATRICULA;
		END
	ELSE
		BEGIN
			IF(@HORAS < 1230)
				BEGIN
					UPDATE CAT_ALUMNOS SET CAT_ALU_STATUS = 1 WHERE
CAT_ALU_MATRICULA = @MATRICULA;
				END
			ELSE
				BEGIN
					UPDATE CAT_ALUMNOS SET CAT_ALU_STATUS = 2 WHERE
CAT_ALU_MATRICULA = @MATRICULA;
				END
		END
END
```

7) Cada alumno tiene un tutor, crear un programa (T-SQL) que al ejecutarse guarde un histórico de los tutores actuales, y sean reasignados a diferentes alumnos, considerando que el proceso se ejecutará por semestre. No se debe contemplar en este proceso a los alumnos con horas COMPLETAS.

```
CREATE OR ALTER TRIGGER TG_HISTORICO
ON HIST_SEMESTRES
FOR INSERT
AS
BEGIN
	DECLARE @ID INTEGER = (SELECT inserted.HIST_SEM_ID - 1 FROM inserted);
	DECLARE @ACTUAL INTEGER = (SELECT inserted.HIST_SEM_ID FROM inserted);
	INSERT INTO
HIST_TUTORES(HIST_TUT_ID,HIST_TUT_MATRICULA,HIST_TUT_SEMESTRE,HIST_TUT_EVENTO,HIST_TU
T_HORAS)
	SELECT CA.CAT_ALU_TUTOR,
	CA.CAT_ALU_MATRICULA,
	@ID,
	ISNULL(HIST_OBD_EVENTO,0),
	ISNULL(SUM((DATEDIFF(HOUR,HIST_OBD_ENTRADA,HIST_OBD_SALIDA))),0)
FROM HIST_DETALLE RIGHT JOIN CAT_ALUMNOS CA ON CA.CAT_ALU_MATRICULA =
HIST_OBD_MATRICULA
AND CA.CAT_ALU_STATUS <2
GROUP BY CA.CAT_ALU_MATRICULA,HIST_OBD_EVENTO,CA.CAT_ALU_TUTOR
```

8) Cada vez que se registre un nuevo alumno se le debe asignar una matrícula, la cual es secuencial y lleva la siguiente estructura:

A182021292

- A o B dependiendo el semestre que se registro
- Año de ingreso a 2 dígitos
- Clave de escuela (20)
- Dígito de turno inicial (matutino 1, vespertino 2)
- Número secuencial que abarque 4 cifras }

```
CREATE SEQUENCE dbo.ALUMNOSECUENCIA
	AS INT
		START WITH 1
		INCREMENT BY 1
		MINVALUE 1 MAXVALUE 10000
		CYCLE;
CREATE OR ALTER PROCEDURE SP_MATRICULA @TURNO INT, @MATRICULA NVARCHAR(15) OUTPUT
AS
BEGIN
	SELECT @MATRICULA = CONCAT((SELECT HGD.HIST_GED_GENERACION FROM
HIST_GENERACION_DETALLE HGD INNER JOIN
HIST_SEMESTRES HS ON HS.HIST_SEM_ID = HGD.HIST_GED_SEMESTRE AND HS.HIST_SEM_STATUS =
1),
SUBSTRING(CAST(YEAR(GETDATE()) AS NVARCHAR),3,LEN(CAST(YEAR(GETDATE()) AS
NVARCHAR))),'20',CAST(@TURNO AS NVARCHAR),REPLACE(STR(CAST(NEXT VALUE FOR
dbo.ALUMNOSECUENCIA AS NVARCHAR), 4),' ','0'));
END

BEGIN
DECLARE @MATRICULA NVARCHAR(15);
EXEC SP_MATRICULA 1, @MATRICULA OUTPUT;
PRINT @MATRICULA;
END
```

8.- Del ejercicio creado en el punto anterior realizar las siguientes

consultas

a) Cantidad de alumnos que tiene cada tutor por semestre

```
SELECT CA.CAT_ALU_TUTOR TUTOR,COUNT(CA.CAT_ALU_MATRICULA) ALUMNOS FROM CAT_ALUMNOS CA
WHERE CA.CAT_ALU_STATUS <> 4 AND (SELECT COUNT(*) FROM HIST_SEMESTRES HS WHERE
GETDATE()
BETWEEN HS.HIST_SEM_APERTURA AND HS.HIST_SEM_CIERRE) >= 1

GROUP BY CA.CAT_ALU_TUTOR;
```

b) Lista de los alumnos que se asignaron a un tutor en un semestre determinado

```
SELECT CA.CAT_ALU_TUTOR TUTOR,COUNT(CA.CAT_ALU_MATRICULA) ALUMNOS FROM CAT_ALUMNOS CA
WHERE CA.CAT_ALU_STATUS <> 4 AND (SELECT COUNT(*) FROM HIST_SEMESTRES HS WHERE
HS.HIST_SEM_STATUS = 1 AND GETDATE()
BETWEEN HS.HIST_SEM_APERTURA AND HS.HIST_SEM_CIERRE) >= 1

GROUP BY CA.CAT_ALU_TUTOR;
```

c) Cantidad de alumnos con horas completas, cantidad de alumnos con horas incompletas y cantidad de alumnos con ninguna hora por tutor y por semestre.

```
WITH SEARCH AS(
SELECT DISTINCT(HT.HIST_TUT_MATRICULA) "MATRICULA", 'INCOMPLETAS' "TIPO",
HT.HIST_TUT_SEMESTRE FROM HIST_TUTORES HT WHERE HT.HIST_TUT_HORAS > 0
UNION SELECT DISTINCT(HT.HIST_TUT_MATRICULA) "MATRICULA", 'SIN HORAS' "TIPO",
HT.HIST_TUT_SEMESTRE FROM HIST_TUTORES HT WHERE HT.HIST_TUT_HORAS = 0
)
SELECT SEARCH.TIPO, COUNT(SEARCH.MATRICULA) "CUANTOS"  FROM SEARCH GROUP BY
SEARCH.MATRICULA, SEARCH.TIPO, SEARCH.HIST_TUT_SEMESTRE

UNION
SELECT 'COMPLETAS' "STATUS", COUNT(CA.CAT_ALU_MATRICULA) "CUANTOS" FROM CAT_ALUMNOS
CA
WHERE CA.CAT_ALU_MATRICULA NOT IN(SELECT HT.HIST_TUT_MATRICULA FROM HIST_TUTORES HT)
AND

CA.CAT_ALU_STATUS = 2 GROUP BY CA.CAT_ALU_MATRICULA;
```

d) Lista por alumno de los tutores que se le han asignado por semestre (solo un registro por alumno)

```
SELECT HT.HIST_TUT_ID, HT.HIST_TUT_SEMESTRE, COUNT(HT.HIST_TUT_MATRICULA) "CUANTOS"
FROM HIST_TUTORES HT

GROUP BY HT.HIST_TUT_MATRICULA,HT.HIST_TUT_ID, HT.HIST_TUT_SEMESTRE;
```

8. Cree una base de datos con un tamaño definido de 1GB para la parte lógica y física. Que se guarde en la ubicación de mis archivos

Nota: Así quedaría, pero recordemos que dará un error por falta de credenciales (permisos) para poder escribir los archivos de base de datos.

```
CREATE DATABASE [BASE_EXAMEN]
ON PRIMARY
(NAME = N'BASE_EXAMEN', FILENAME = N'C:\Program Files\Mis Archivos\BASE_EXAMEN.mdf',
SIZE = 1GB,
MAXSIZE = UNLIMITED, FILEGROWTH = 500MB)
LOG ON
(NAME = N'BASE_EXAMEN_log', FILENAME = N'C:\Program Files\Mis
Archivos\BASE_EXAMEN_log.ldf', SIZE = 1GB, MAXSIZE = UNLIMITED, FILEGROWTH = 500MB)
```

9. Crear un ejemplo de snapshot de base de datos.

Nota: Una vez, pensando que se pudo escribir la base de datos el querie quedaría así.

```
USE BASE_EXAMEN;
CREATE DATABASE BASE_EXAMEN_SNAPSHOT
ON
(NAME = N'BASE_EXAMEN_SNAPSHOT',
FILENAME = N'C:\Program Files\Mis Archivos\BASE_EXAMEN_SNAPSHOT.ss')
    AS SNAPSHOT OF [BASE_EXAMEN_SNAPSHOT];
```

10.- Crea un programa (TL-SQL) que genera la siguiente salida.

```
begin
       declare @ARBOL int = 6;
       select '*'
       union all
       select REPLICATE('*', f.number) from dbo.fn_Fibonacci(@ARBOL) f where f.number
> 1;
end
CREATE FUNCTION fn_Fibonacci(@max int)
RETURNS @numbers TABLE(number int)
```

```
AS
BEGIN
	Declare @n1 int = 0,@n2 int =1,@i int=0,@temp int
	Insert Into @numbers Values(@n1),(@n2)
	WHILE (@i<=@max-2)
	BEGIN
		Insert Into @numbers Values(@n2+@n1)
		set @temp = @n2
		Set @n2 = @n2 + @n1
		Set @n1 = @temp
		Set @i += 1
	END
	RETURN
END
```