

UNIVERSIDAD PRIVADA-DE-TACNA



INGENIERIA DE SISTEMAS

TITULO:

Laboratorio No 02

CURSO:

BASE DE DATOS II

DOCENTE(ING):

Patrick Cuadros Quiroga

Integrantes:

Andrés Eduardo de la Barra Vásquez	(2016055087)
David Damian Mamani	(2016055194)
Andre Sebastian Reinoso Aranda	(2016055275)
M	(201)

Índice

1. INFORMACIÓN GENERALI	1
1.1. Objetivos:	1
1.2. Equipos, materiales, programas y recursos utilizados:	1
2. Marco teorico	2
3. Desarrollo	3
4. Conclusiones	9

1. INFORMACIÓN GENERAL

1.1. Objetivos:

- Conocer los metodos de prueba para la base de datos dada.
- Atachar una base de datos.
- Visualiza en el gestor de BD las sentencias generadas .

1.2. Equipos, materiales, programas y recursos utilizados:

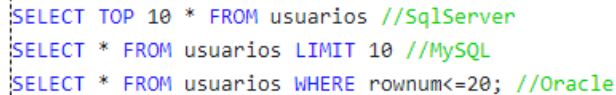
- Microsoft SQL Server 2016.
- Computadora con sistema operativo Windows 10.
- Base de datos SchoolDB
- Microsoft Visual Studio 2017
- Una cuenta en Github para subir los cambios

2. Marco teorico

¿Qué es un ORM?

- El Mapeo Objeto-Relacional o ORM es un modelo de programacion que consiste en la transformar la logica de programacion (entidades) en tablas, simplificando las tareas como son el modelado.

La ventaja mas certeras en el ORM es la capacidad de poder migrar la DB sin la necesidad de saber todo el entorno completo ya que se trabajaria con un lenguaje unificado.



```
SELECT TOP 10 * FROM usuarios //SqlServer
SELECT * FROM usuarios LIMIT 10 //MySQL
SELECT * FROM usuarios WHERE rownum<=20; //Oracle
```

Mismo proposito diferentes formas de hacerlo.

```
var items = context.ProductoSet.OrderByDescending(u => u.Producto.Count).Take(5);
```

En este caso la anterior linea de codigo devuelve lo mismo que la primera imagen.

3. Desarrollo

Ejercicio 1: Obtener al estudiante con el ID Uno

– Código

Mediante el código se podrá buscar en la base de datos (Tabla Estudiante) al estudiante siempre y cuando su nombre sea “Bill”

```
0 references | 0 changes | 0 authors, 0 changes
public void BuscarAlPrimerEstudianteConElNombreBill()
{
    using (var ctx = new SchoolContext())
    {
        var student = (from s in ctx.Students
                        where s.StudentName == "Bill"
                        select s).FirstOrDefault<Student>();
    }
}
```

– Gestor de base de Datos

Esta sentencia que es generada por el código en el gestor de la base de datos muestra como efectúa la selección del código en la base de datos el cual nos mostrara el estudiante con el nombre “Bill”

```
ELECT TOP (1)
  [Extent1].[StudentID] AS [StudentID],
  [Extent1].[StudentName] AS [StudentName],
  [Extent1].[DateOfBirth] AS [DateOfBirth],
  [Extent1].[Height] AS [Height],
  [Extent1].[Weight] AS [Weight],
  [Extent1].[RowVersion] AS [RowVersion],
  [Extent1].[GradeId] AS [GradeId]
FROM [dbo].[Students] AS [Extent1]
WHERE N'Bill' = [Extent1].[StudentName]
```

Ejercicio 2: Buscar Estudiantes Agrupados Por Grado

– Código

Consiste en que el código buscara al estudiante dependiendo al grado, es decir que evaluara a cada estudiante con los datos ingresados en la tabla grade

```
✓ | 0 references | 0 changes | 0 authors, 0 changes
public void BuscarEstudiantesAgrupadosPorEstandar()
{
    using (var ctx = new SchoolContext())
    {
        var students = ctx.Students.GroupBy(s => s.Grade);
        foreach (var groupItem in students)
        {
            Console.WriteLine(groupItem.Key);
            foreach (var stud in groupItem)
            {
                Console.WriteLine(stud.StudentID);
            }
        }
    }
}
```

– Gestor de base de Datos

La sentencia generada seleccionara la tabla “Grade” y la tabla “Student”, seguidamente tomara una variable “Distinct1” el cual tomara los valores de la tabla “Grade” y por otro lado la variable “Extent3” tomara los valores de la tabla “Student”, Después lo evaluara para mostrar los valores correspondientes.

```
SELECT
    [Project2].[C1] AS [C1],
    [Project2].[GradeID] AS [GradeID],
    [Project2].[GradeName] AS [GradeName],
    [Project2].[Section] AS [Section],
    [Project2].[C2] AS [C2],
    [Project2].[StudentID] AS [StudentID],
    [Project2].[StudentName] AS [StudentName],
    [Project2].[DateOfBirth] AS [DateOfBirth],
    [Project2].[Height] AS [Height],
    [Project2].[Weight] AS [Weight],
    [Project2].[RowVersion] AS [RowVersion],
    [Project2].[GradeID1] AS [GradeID1]
FROM
    (SELECT
        [Distinct1].[GradeID] AS [GradeID],
        [Distinct1].[GradeName] AS [GradeName],
        [Distinct1].[Section] AS [Section],
        1 AS [C1],
        [Extent3].[StudentID] AS [StudentID],
        [Extent3].[StudentName] AS [StudentName],
        [Extent3].[DateOfBirth] AS [DateOfBirth],
        [Extent3].[Height] AS [Height],
        [Extent3].[Weight] AS [Weight],
        [Extent3].[RowVersion] AS [RowVersion],
        [Extent3].[GradeID] AS [GradeID1],
        CASE WHEN ([Extent3].[StudentID] IS NULL) THEN CAST(NULL AS int) ELSE 1 END AS [C2]
    FROM
        (SELECT DISTINCT
            [Extent2].[GradeID] AS [GradeID],
            [Extent2].[GradeName] AS [GradeName],
            [Extent2].[Section] AS [Section]
        FROM
            [dbo].[Students] AS [Extent1]
        LEFT OUTER JOIN [dbo].[Grades] AS [Extent2] ON ([Distinct1].[GradeID] = [Extent1].[GradeID]) OR ([Distinct1].[GradeID] IS NULL) AND ([Extent1].[GradeID] IS NULL))
    AS [Project2]
    LEFT OUTER JOIN [dbo].[Students] AS [Extent3] ON ([Distinct1].[GradeID] = [Extent3].[GradeID]) OR ([Distinct1].[GradeID] IS NULL) AND ([Extent3].[GradeID] IS NULL))
    AS [Project2]
ORDER BY [Project2].[GradeID] ASC, [Project2].[GradeName] ASC, [Project2].[Section] ASC, [Project2].[C2] ASC
```

Ejercicio 3: Obtener listado de estudiantes ordenados por nombre

– Código

El siguiente código ordenará la tabla “Student” por dependiendo del nombre, es decir, que seguirá el orden alfabético.

```
0 references | 0 changes | 0 authors, 0 changes
public void ObetenerListadoDeEstudiantesOrdenadosPorNombre()
{
    using (var ctx = new SchoolContext())
    {
        var students = ctx.Students.OrderBy(s => s.StudentName).ToList();
        // or descending order
        var descStudents = ctx.Students.OrderByDescending(s => s.StudentName)
            .ToList();
    }
}
```

– Gestor de base de Datos

La sentencia tomará los valores de la tabla estudiante en “Extent1” de tal modo ordenada la posición de ella mediante el “Order By” que estará sujeto a “StudentName” con la propiedad

```
SELECT
    [Extent1].[StudentID] AS [StudentID],
    [Extent1].[StudentName] AS [StudentName],
    [Extent1].[DateOfBirth] AS [DateOfBirth],
    [Extent1].[Height] AS [Height],
    [Extent1].[Weight] AS [Weight],
    [Extent1].[RowVersion] AS [RowVersion],
    [Extent1].[GradeId] AS [GradeId]
FROM [dbo].[Students] AS [Extent1]
ORDER BY [Extent1].[StudentName] DESC
```

Ejercicio 4: Insertar estudiantes satisfactoriamente

- Código

Consiste en que el código buscara al estudiante dependiendo al grado, es decir que evaluara a cada estudiante con los datos ingresados en la tabla "Grade"

```
public void InsertarEstudianteSatisfactoriamente()
{
    using (var context = new SchoolContext())
    {
        var std = new Student()
        {
            StudentName = "Bill"
        };
        context.Students.Add(std);
        context.SaveChanges();
    }
}
```

- Gestor de base de Datos

La sentencia generada seleccionara la tabla "Grade" y la tabla "Student", seguidamente tomara una variable "Distinct1" el cual tomara los valores de la tabla "Grade" y por otro lado la variable "Extent3" tomara los valores de la tabla "Student", Después lo evaluara para mostrar los valores correspondientes.

```
exec sp_executesql N'INSERT [dbo].[Students]([StudentName], [DateOfBirth], [Height], [Weight], [RowVersion], [GradeId])
VALUES (@0, NULL, @1, @2, NULL, NULL)
SELECT [StudentID]
FROM [dbo].[Students]
WHERE @@ROWCOUNT > 0 AND [StudentID] = scope_identity()',N'@0 nvarchar(50),@1 decimal(18,2),@2 real',@0=N'Bill',@1=0,@2=0
```


Ejercicio 5: Eliminar El Primer Grade Satisfactoriamente

– Código

En el siguiente código Eliminará al primero estudiante.

```
using (var context = new SchoolDBEntities())
{
    var std = context.Students.First<Student>();
    context.Students.Remove(std);
    context.SaveChanges();
}
```

– Gestor de base de Datos

En la sentencia seleccionar la tabla “Student”, seguidamente eliminará el Student con el código correspondiente que en este caso será “1”

```
exec sp_executesql N'DELETE [dbo].[Grades]
WHERE ([GradeId] = @0)',N'@0 int',@0=1
```

Ejercicio 6: Agregar Tres Grades Satisfactoriamente

– Código

El siguiente código evaluará la lista “Grades” de tal modo que insertará nuevos parámetros o valores a la lista. Seguidamente agregará los datos y guardará cambios

```
public void AgregarTresGradesSatisfactoriamente()
{
    IList<Grade> newStudents = new List<Grade>() {
        new Grade() { GradeName = "Steve" },
        new Grade() { GradeName = "Bill" },
        new Grade() { GradeName = "James" },
    };

    using (var context = new SchoolContext())
    {
        context.Grades.AddRange(newStudents);
        context.SaveChanges();
    }
}
```

– Gestor de base de Datos

La sentencia insertará los parámetros establecidos en el código, de tal modo que seleccionará la tabla correspondiente e insertará los datos.

```
exec sp_executesql N'INSERT [dbo].[Grades]([GradeName], [Section])
VALUES (@0, NULL)
SELECT [GradeId]
FROM [dbo].[Grades]
WHERE @@ROWCOUNT > 0 AND [GradeId] = scope_identity()', N'@0 nvarchar(max) ', @0=N'Steve'
```

```
exec sp_executesql N'INSERT [dbo].[Grades]([GradeName], [Section])
VALUES (@0, NULL)
SELECT [GradeId]
FROM [dbo].[Grades]
WHERE @@ROWCOUNT > 0 AND [GradeId] = scope_identity()', N'@0 nvarchar(max) ', @0=N'Bill'
```

```
exec sp_executesql N'INSERT [dbo].[Grades]([GradeName], [Section])
VALUES (@0, NULL)
SELECT [GradeId]
FROM [dbo].[Grades]
WHERE @@ROWCOUNT > 0 AND [GradeId] = scope_identity()', N'@0 nvarchar(max) ', @0=N'James'
```

4. Conclusiones

Desarrollo

- Items
Es correcta

	last_name	job_id	Sal
1	King	AD_PRES	24000.00
2	Kochhar	AD_VP	17000.00
3	De Haan	AD_VP	17000.00
4	Hunold	IT_PROG	9000.00
5	Emst	IT_PROG	6000.00
6	Austin	IT_PROG	4800.00
7	Pataballa	IT_PROG	4800.00
8	Lorentz	IT_PROG	4200.00
9	Greenberg	FI_MGR	12008.00
10	Faviet	FI_ACCOUNT	9000.00
11	Chen	FI_ACCOUNT	8200.00
12	Sciarra	FI_ACCOUNT	7700.00