Enunciado:

Tema: Bases de datos y ADO con procedimientos almacenados (**stored procedure**) en una base SQL Server.

Nuestro objetivo será migrar la aplicación de biblioteca para que en lugar de funcionar con sentencias de **sql embebido** en la aplicación, lo haga con invocaciones a **stored procedure**, podemos ver en la **Figura 1** la comparación y por ende las diferencias entre ambos códigos.

Para que el desarrollador tome como código de patrón se le proporcionara el persistidor "**LectorPersistidor.cs**" ya desarrollado invocando stored procedure y debemos entonces desarrollar el persistidor "**LibroPersistidor.cs**" que sigue escrito con "SQL embebido".

Figura 1

SQL embebido

```
private void Update(Lector lector)
{
    string connectionString = ConfigurationManager.ConnectionStrings["c
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();
        const string sqlQuery =
            @"  UPDATE Lector
                SET
                    dni = @dni,
                    apellido = @apellido
                WHERE id_lector = @id_lector";
        using (SqlCommand cmd = new SqlCommand(sqlQuery, cnx))
        {



            cmd.Parameters.AddWithValue("@dni", lector.DNI);
            cmd.Parameters.AddWithValue("@apellido", lector.Apellido);
            cmd.Parameters.AddWithValue("@id_lector", lector.Id);

            int count = cmd.ExecuteNonQuery();
            if (count != 1) throw new Exception("La entidad no pudo ser
        }
    }
}
```

Invocación a stored procedure

```
private void Update(Lector lector)
{
    string connectionString = ConfigurationManager.ConnectionStrings["cnnS
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();









        using (SqlCommand cmd = new SqlCommand())
        {
            cmd.Connection = cnx;
            cmd.CommandText = "lector_upd";
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@dni", lector.DNI);
            cmd.Parameters.AddWithValue("@apellido", lector.Apellido);
            cmd.Parameters.AddWithValue("@id_lector", lector.Id);

            int count = cmd.ExecuteNonQuery();
            if (count != 1) throw new Exception("La entidad no pudo ser mo
        }
    }
}
```

La **Figura 2**, muestra el código del stored procedure "**lector_upd**" escrito en lenguaje Transact-SQL (T-SQL) y almacenado en una base SQL Server, este será invocado desde nuestra aplicación escrita en código C#, como podemos observar en la **Figura 1**.

Figura 2

```sql
/***
* Descripción: Modifica un registro de la tabla Lector.
***/
CREATE PROCEDURE lector_upd
    @dni int = 0,
    @apellido varchar(50),
    @id_lector int
AS
BEGIN
    UPDATE Lector
    SET
        dni = @dni,
        apellido = @apellido
    WHERE id_lector = @id_lector
END
```

Figura 3	
Nombre del "stored procedure"	Descripción
lector_upd	Modifica un registro de la tabla Lector.
lector_ins	Adiciona un registro en la tabla Lector.
lector_del	Elimina un registro por su clave primaria (primary key)
lector_sel_all	Selecciona todos los registros de la tabla Lector.
lector_sel_by_id	Selecciona un registro de la tabla Lector por su clave primaria (primary key).
lector_sel_max_id_lector	Selecciona el máximo valor del campo clave primaria (primary key).

La **Figura 4**, muestra el listado de los "stored procedure" necesario para realizar el CRUD (**C**reate, **R**ead, **U**pdate, **D**elete) sobre la tabla "**lector**", queda en manos del desarrollador implementar los respectivos sobre la tabla "**libro**".

Figura 4

```
/***
* Descripción: Modifica un registro de la tabla Lector.
***/
CREATE PROCEDURE lector_upd
    @dni int = 0,
    @apellido varchar(50),
    @id_lector int
AS
BEGIN
    UPDATE Lector
    SET
        dni = @dni,
        apellido = @apellido
    WHERE id_lector = @id_lector
END
```

```
/***
* Descripción: Selecciona el máximo valor del campo clave primaria (primary key).
***/
CREATE PROCEDURE lector_sel_max_id_lector
AS
BEGIN
    SELECT MAX(id_lector) FROM Lector
END
```

```
/***
* Descripción: Selecciona un registro de la tabla Lector por su clave primaria (primary key).
***/
CREATE PROCEDURE [lector_sel_by_id]
    @id_lector int
AS
BEGIN
    SELECT * FROM Lector WHERE id_lector = @id_lector
END
```

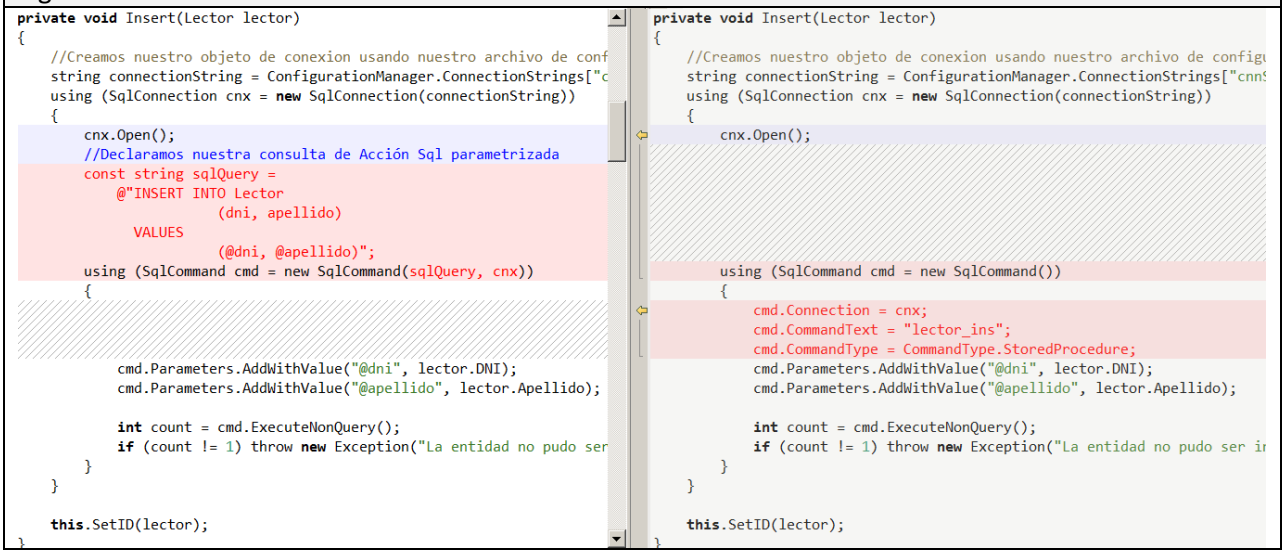
```
/***
* Descripción: Selecciona todos los registros de la tabla Lector.
***/
CREATE PROCEDURE lector_sel_all
AS
BEGIN
    SELECT * FROM Lector ORDER BY id_lector ASC
END
```

```
/***
* Descripción: Adiciona un registro en la tabla Lector.
***/
CREATE PROCEDURE lector_ins
    @dni int = 0,
    @apellido varchar(50)
AS
BEGIN
    INSERT INTO Lector
            (dni, apellido)
    VALUES
            (@dni, @apellido)
END
```

```
/***
* Descripción: Elimina un registro por su clave primaria (primary key)
***/
CREATE PROCEDURE lector_del
    @id_lector int
AS
BEGIN
    DELETE FROM Lector WHERE id_lector = @id_lector
END
```

El siguiente listado de figuras muestra las diferencias existentes entre "**SQL embebido**" y "**stored procedure**", mediante la comparación de sus códigos respectivos código.

Figura 5



```
private void Insert(Lector lector)
{
    //Creamos nuestro objeto de conexion usando nuestro archivo de conf
    string connectionString = ConfigurationManager.ConnectionStrings["c
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();
        //Declaramos nuestra consulta de Acción Sql parametrizada
        const string sqlQuery =
            @"INSERT INTO Lector
                        (dni, apellido)
              VALUES
                        (@dni, @apellido)";
        using (SqlCommand cmd = new SqlCommand(sqlQuery, cnx))
        {

            cmd.Parameters.AddWithValue("@dni", lector.DNI);
            cmd.Parameters.AddWithValue("@apellido", lector.Apellido);

            int count = cmd.ExecuteNonQuery();
            if (count != 1) throw new Exception("La entidad no pudo ser
        }
    }

    this.SetID(lector);
}
```

```
private void Insert(Lector lector)
{
    //Creamos nuestro objeto de conexion usando nuestro archivo de configu
    string connectionString = ConfigurationManager.ConnectionStrings["cnn
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();

        using (SqlCommand cmd = new SqlCommand())
        {
            cmd.Connection = cnx;
            cmd.CommandText = "lector_ins";
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@dni", lector.DNI);
            cmd.Parameters.AddWithValue("@apellido", lector.Apellido);

            int count = cmd.ExecuteNonQuery();
            if (count != 1) throw new Exception("La entidad no pudo ser in
        }
    }

    this.SetID(lector);
}
```

Figura 6

```
private void Update(Lector lector)
{
    string connectionString = ConfigurationManager.ConnectionStrings["c
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();
        const string sqlQuery =
            @"  UPDATE Lector
                SET
                    dni = @dni,
                    apellido = @apellido
                WHERE id_lector = @id_lector";
        using (SqlCommand cmd = new SqlCommand(sqlQuery, cnx))
        {



            cmd.Parameters.AddWithValue("@dni", lector.DNI);
            cmd.Parameters.AddWithValue("@apellido", lector.Apellido);
            cmd.Parameters.AddWithValue("@id_lector", lector.Id);

            int count = cmd.ExecuteNonQuery();
            if (count != 1) throw new Exception("La entidad no pudo ser
        }
    }
}
```

```
private void Update(Lector lector)
{
    string connectionString = ConfigurationManager.ConnectionStrings["cnn
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();









        using (SqlCommand cmd = new SqlCommand())
        {
            cmd.Connection = cnx;
            cmd.CommandText = "lector_upd";
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@dni", lector.DNI);
            cmd.Parameters.AddWithValue("@apellido", lector.Apellido);
            cmd.Parameters.AddWithValue("@id_lector", lector.Id);

            int count = cmd.ExecuteNonQuery();
            if (count != 1) throw new Exception("La entidad no pudo ser m
        }
    }
}
```

Figura 7

```
private void SetID(Lector lector)
{
    //Creamos nuestro objeto de conexion usando nuestro archivo de conf
    string connectionString = ConfigurationManager.ConnectionStrings["c
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        //Declaramos nuestra consulta de Acción Sql parametrizada
        const string sqlQuery = @"SELECT MAX(id_lector) FROM Lector";
        using (SqlCommand cmd = new SqlCommand(sqlQuery, cnx))
        {



            DataTable dataTable = new DataTable();
            SqlDataAdapter dataAdapter = new SqlDataAdapter(cmd);
            dataAdapter.Fill(dataTable);
            if (dataTable.Rows[0][0] != DBNull.Value)
            {
                lector.Id = Convert.ToInt32(dataTable.Rows[0][0]);
            }
        }
    }
}
```

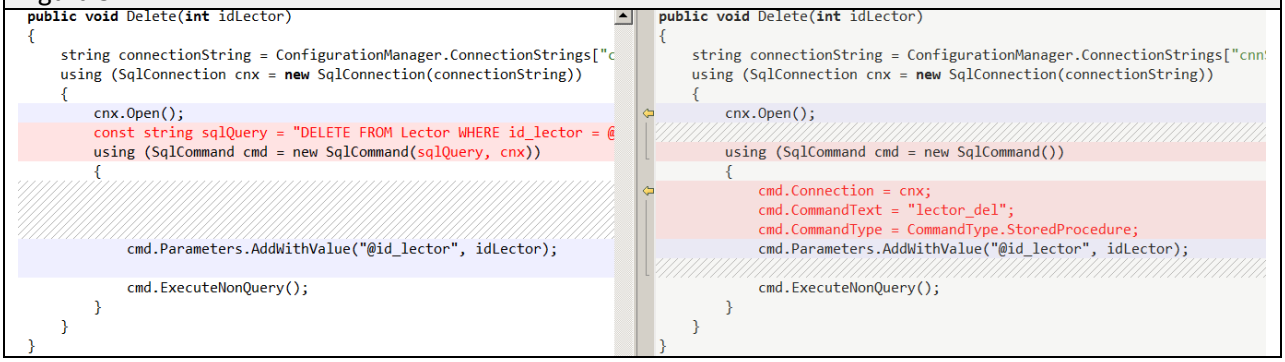
```
private void SetID(Lector lector)
{
    //Creamos nuestro objeto de conexion usando nuestro archivo de config
    string connectionString = ConfigurationManager.ConnectionStrings["cnn
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {



        using (SqlCommand cmd = new SqlCommand())
        {
            cmd.Connection = cnx;
            cmd.CommandText = "lector_sel_max_id_lector";
            cmd.CommandType = CommandType.StoredProcedure;

            DataTable dataTable = new DataTable();
            SqlDataAdapter dataAdapter = new SqlDataAdapter(cmd);
            dataAdapter.Fill(dataTable);
            if (dataTable.Rows[0][0] != DBNull.Value)
            {
                lector.Id = Convert.ToInt32(dataTable.Rows[0][0]);
            }
        }
    }
}
```

Figura 8

```
public Lector GetByid(int idLector)
{
    Lector lector = null;
    string connectionString = ConfigurationManager.ConnectionStrings["c
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        const string sqlQuery = "SELECT * FROM Lector WHERE id_lector =
        using (SqlCommand cmd = new SqlCommand(sqlQuery, cnx))
        {



            cmd.Parameters.AddWithValue("@id_lector", idLector);

            DataTable table = new DataTable();
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(table);
            if (table.Rows.Count != 0)
            {
                DataRow row = table.Rows[0];
                lector = new Lector
                {
                    Id = Convert.ToInt32(row["id_lector"]),
                    DNI = Convert.ToInt32(row["dni"]),
                    Apellido = Convert.ToString(row["apellido"])
                };
            }
        }
```

```
public Lector GetByid(int idLector)
{
    Lector lector = null;
    string connectionString = ConfigurationManager.ConnectionStrings["cnn
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {



        using (SqlCommand cmd = new SqlCommand())
        {
            cmd.Connection = cnx;
            cmd.CommandText = "lector_sel_by_id";
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@id_lector", idLector);

            DataTable table = new DataTable();
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(table);
            if (table.Rows.Count != 0)
            {
                DataRow row = table.Rows[0];
                lector = new Lector
                {
                    Id = Convert.ToInt32(row["id_lector"]),
                    DNI = Convert.ToInt32(row["dni"]),
                    Apellido = Convert.ToString(row["apellido"])
                };
            }
        }
```

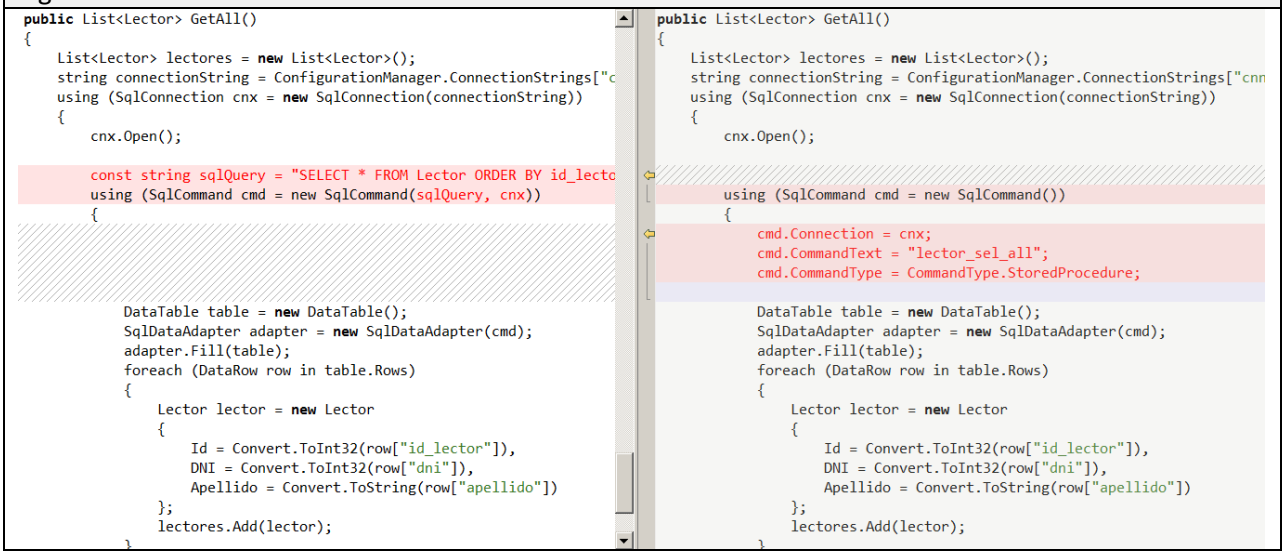
Figura 9



```
public void Delete(int idLector)
{
    string connectionString = ConfigurationManager.ConnectionStrings["c
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();
        const string sqlQuery = "DELETE FROM Lector WHERE id_lector = @
        using (SqlCommand cmd = new SqlCommand(sqlQuery, cnx))
        {
            cmd.Parameters.AddWithValue("@id_lector", idLector);

            cmd.ExecuteNonQuery();
        }
    }
}
```

```
public void Delete(int idLector)
{
    string connectionString = ConfigurationManager.ConnectionStrings["cnn
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();

        using (SqlCommand cmd = new SqlCommand())
        {
            cmd.Connection = cnx;
            cmd.CommandText = "lector_del";
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@id_lector", idLector);

            cmd.ExecuteNonQuery();
        }
    }
}
```

Figura 10



```
public List<Lector> GetAll()
{
    List<Lector> lectores = new List<Lector>();
    string connectionString = ConfigurationManager.ConnectionStrings["c
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();

        const string sqlQuery = "SELECT * FROM Lector ORDER BY id_lecto
        using (SqlCommand cmd = new SqlCommand(sqlQuery, cnx))
        {
            DataTable table = new DataTable();
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(table);
            foreach (DataRow row in table.Rows)
            {
                Lector lector = new Lector
                {
                    Id = Convert.ToInt32(row["id_lector"]),
                    DNI = Convert.ToInt32(row["dni"]),
                    Apellido = Convert.ToString(row["apellido"])
                };
                lectores.Add(lector);
            }
```

```
public List<Lector> GetAll()
{
    List<Lector> lectores = new List<Lector>();
    string connectionString = ConfigurationManager.ConnectionStrings["cnn
    using (SqlConnection cnx = new SqlConnection(connectionString))
    {
        cnx.Open();

        using (SqlCommand cmd = new SqlCommand())
        {
            cmd.Connection = cnx;
            cmd.CommandText = "lector_sel_all";
            cmd.CommandType = CommandType.StoredProcedure;

            DataTable table = new DataTable();
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(table);
            foreach (DataRow row in table.Rows)
            {
                Lector lector = new Lector
                {
                    Id = Convert.ToInt32(row["id_lector"]),
                    DNI = Convert.ToInt32(row["dni"]),
                    Apellido = Convert.ToString(row["apellido"])
                };
                lectores.Add(lector);
            }
```