CRUD Operations Using Connected Architecture (ADO.NET)

CRUD stands for:

Create → INSERT

Read → SELECT

Update → UPDATE

Delete → DELETE

In Connected Architecture, all CRUD operations are performed using the same fundamental pipeline:

SqlConnection → SqlCommand → (optional) SqlDataReader

The difference lies only in:

the SQL statement

the execution method used on SqlCommand

Let us now validate each CRUD operation one by one, both conceptually and practically.

1. CREATE Operation (INSERT) – ✅ COMPLETED

Theory

The create operation inserts new data into the database.
In connected architecture, INSERT statements do not return rows, so they are executed using:

```
ExecuteNonQuery()
```

The database performs the insertion and returns only the number of rows affected.

Code (Connected Architecture)

```
string query =
"INSERT INTO Students (Name, Age) VALUES
(@name, @age)";
```

```
using (SqlConnection connection = new
SqlConnection(connectionString))
{
connection.Open();

SqlCommand command = new
SqlCommand(query, connection);
command.Parameters.AddWithValue("@name",
"Rohit");
command.Parameters.AddWithValue("@age", 21);

int rowsAffected =
command.ExecuteNonQuery();
}
```

✓ Uses live connection

✓ Uses SqlCommand

✓ Uses ExecuteNonQuery

✓ Fully connected architecture

Status: COMPLETE

2. READ operation (SELECT) – ✅ COMPLETED

Theory

The Read operation retrieves data from the database.
In connected architecture, SELECT queries return result sets, which must be read sequentially using:

SqlDataReader via ExecuteReader()

The connection remains open while data is being read.

Code (Connected Architecture)

```
string query = "SELECT Id, Name, Age FROM Students";

using (SqlConnection connection = new SqlConnection(connectionString))
{
connection.Open();

SqlCommand command = new
SqlCommand(query, connection);
SqlDataReader reader =
command.ExecuteReader();

while (reader.Read())
{
Console.WriteLine(reader["Name"]);
}

reader.close();
}
```

✓ Forward-only
✓ Read-only
✓ Connection-dependent

Status: COMPLETE

3. UPDATE operation (UPDATE) – ⚠️
CONCEPTUALLY COVERED, NOW FORMALLY CLOSED
Theory

The Update operation modifies existing records.

Like INSERT, UPDATE statements:

do not return result sets

only report how many rows were affected

Hence, they use:

ExecuteNonQuery()

Code (Connected Architecture)

```
string query =
"UPDATE Students SET Age = @age WHERE
Name = @name";

using (SqlConnection connection = new
SqlConnection(connectionString))
{
connection.Open();

SqlCommand command = new
SqlCommand(query, connection);
command.Parameters.AddWithValue("@age",
23);
command.Parameters.AddWithValue("@name",
"Rohit");

int rowsAffected =
command.ExecuteNonQuery();
}
```

✓ Same pipeline as INSERT
✓ Connected architecture
✓ Uses parameters (secure)

Status: NOW COMPLETE

4. DELETE operation (DELETE) – ⚠️
CONCEPTUALLY COVERED, NOW FORMALLY CLOSED
Theory

The Delete operation removes records from the database.
DELETE statements:

modify database state

return no rows

Hence, again:

ExecuteNonQuery()

Code (Connected Architecture)

```
string query =
"DELETE FROM Students WHERE Name =
@name";

using (SqlConnection connection = new
SqlConnection(connectionString))
{
connection.Open();

SqlCommand command = new
SqlCommand(query, connection);
command.Parameters.AddWithValue("@name",
"Rohit");

int rowsAffected =
command.ExecuteNonQuery();
}
```