In **Entity Framework Core** (EF Core 7+), you can process or delete records **without loading them all into memory** by using **streaming** or **bulk operations**.

If you want to **sweep through records one at a time** (e.g., for processing or cleanup) without materializing the entire dataset, you should use `AsNoTracking()` with `AsAsyncEnumerable()` or `IAsyncEnumerable<T>`.

Example: Process Records One at a Time Without Loading All into Memory

```
using System;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;

public class MyEntity
{
    public int Id { get; set; }
    public string Data { get; set; }
}

public class MyDbContext : DbContext
{
    public DbSet<MyEntity> MyEntities { get; set; }
}

public class Program
{
    public static async Task Main()
    {
        using var db = new MyDbContext();
```

```
        // Stream results without tracking to avoid memory bloat
        await foreach (var entity in db.MyEntities
            .AsNoTracking()
            .AsAsyncEnumerable())
        {
            // Process each entity individually
            Console.WriteLine($"Processing ID: {entity.Id}");

            // Example: Delete without loading into memory
            await db.MyEntities
                .Where(e => e.Id == entity.Id)
                .ExecuteDeleteAsync();
        }
    }
}
```

Key Points

- `AsNoTracking()`: Prevents EF from tracking entities, reducing memory usage.
- `AsAsyncEnumerable()`: Streams results instead of buffering them all.
- `ExecuteDeleteAsync()` (EF Core 7+): Deletes directly in the database without loading the entity.

Alternative: Bulk Delete Without Iterating

If you don’t need per-record logic and just want to delete in bulk:

```
await db.MyEntities
    .Where(e => e.Data == "obsolete")
    .ExecuteDeleteAsync();
```

This runs a **single SQL** `DELETE` statement — fastest and most memory-efficient.

When to Use Which

- **Per-record processing** → Use `AsNoTracking()` + `AsAsyncEnumerable()`.
- **Bulk delete/update** → Use `ExecuteDeleteAsync()` / `ExecuteUpdateAsync()`.

If you want, I can give you a **batched sweep** version that processes, say, 500 records at a time for better performance.
Do you want me to prepare that?