Classification of Standard Query Operators by Manner of Execution (C#)

Article • 06/28/2022 • 3 minutes to read • 10 contributors

In this article

The LINQ to Objects implementations of the standard query operator methods execute in one of two main ways: immediate or deferred. The query operators that use deferred execution can be additionally divided into two categories: streaming and non-streaming. If you know how the different query operators execute, it may help you understand the results that you get from a given query. This is especially true if the data source is changing or if you are building a query on top of another query. This topic classifies the standard query operators according to their manner of execution.

Manners of Execution

Immediate

Immediate execution means that the data source is read and the operation is performed once. All the standard query operators that return a scalar result execute immediately. You can force a query to execute immediately using the Enumerable.ToList or Enumerable.ToArray methods. Immediate execution provides reuse of query results, not query declaration. The results are retrieved once, then stored for future use.

Deferred

Deferred execution means that the operation is not performed at the point in the code where the query is declared. The operation is performed only when the query variable is

enumerated, for example by using a `foreach` statement. This means that the results of executing the query depend on the contents of the data source when the query is executed rather than when the query is defined. If the query variable is enumerated multiple times, the results might differ every time. Almost all the standard query operators whose return type is IEnumerable<T> or IOrderedEnumerable<TElement> execute in a deferred manner. Deferred execution provides the facility of query reuse since the query fetches the updated data from the data source each time query results are iterated.

Query operators that use deferred execution can be additionally classified as streaming or non-streaming.

Streaming

Streaming operators do not have to read all the source data before they yield elements. At the time of execution, a streaming operator performs its operation on each source element as it is read and yields the element if appropriate. A streaming operator continues to read source elements until a result element can be produced. This means that more than one source element might be read to produce one result element.

Non-Streaming

Non-streaming operators must read all the source data before they can yield a result element. Operations such as sorting or grouping fall into this category. At the time of execution, non-streaming query operators read all the source data, put it into a data structure, perform the operation, and yield the resulting elements.

Classification Table

The following table classifies each standard query operator method according to its method of execution.

> **Note**
>
> If an operator is marked in two columns, two input sequences are involved in the operation, and each sequence is evaluated differently. In these cases, it is always the first sequence in the parameter list that is evaluated in a deferred, streaming manner.

Standard Query Operator	Return Type	Immediate Execution	Deferred Streaming Execution	Deferred Non-Streaming Execution
Aggregate	TSource	X		
All	Boolean	X		
Any	Boolean	X		
AsEnumerable	IEnumerable<T>		X	
Average	Single numeric value	X		
Cast	IEnumerable<T>		X	
Concat	IEnumerable<T>		X	
Contains	Boolean	X		
Count	Int32	X		
DefaultIfEmpty	IEnumerable<T>		X	
Distinct	IEnumerable<T>		X	
ElementAt	TSource	X		
ElementAtOrDefault	TSource	X		
Empty	IEnumerable<T>	X		
Except	IEnumerable<T>		X	X
First	TSource	X		
FirstOrDefault	TSource	X		
GroupBy	IEnumerable<T>			X
GroupJoin	IEnumerable<T>		X	X
Intersect	IEnumerable<T>		X	X
Join	IEnumerable<T>		X	X
Last	TSource	X		
LastOrDefault	TSource	X		

Standard Query Operator	Return Type	Immediate Execution	Deferred Streaming Execution	Deferred Non-Streaming Execution
LongCount	Int64	X		
Max	Single numeric value, TSource, or TResult	X		
Min	Single numeric value, TSource, or TResult	X		
OfType	IEnumerable<T>		X	
OrderBy	IOrderedEnumerable<TElement>			X
OrderByDescending	IOrderedEnumerable<TElement>			X
Range	IEnumerable<T>		X	
Repeat	IEnumerable<T>		X	
Reverse	IEnumerable<T>			X
Select	IEnumerable<T>		X	
SelectMany	IEnumerable<T>		X	
SequenceEqual	Boolean	X		
Single	TSource	X		
SingleOrDefault	TSource	X		
Skip	IEnumerable<T>		X	
SkipWhile	IEnumerable<T>		X	
Sum	Single numeric value	X		
Take	IEnumerable<T>		X	
TakeWhile	IEnumerable<T>		X	
ThenBy	IOrderedEnumerable<TElement>			X
ThenByDescending	IOrderedEnumerable<TElement>			X
ToArray	TSource array	X		

Standard Query Operator	Return Type	Immediate Execution	Deferred Streaming Execution	Deferred Non-Streaming Execution
ToDictionary	Dictionary<TKey,TValue>	X		
ToList	IList<T>	X		
ToLookup	ILookup<TKey,TElement>	X		
Union	IEnumerable<T>		X	
Where	IEnumerable<T>		X	

See also

- Enumerable
- Standard Query Operators Overview (C#)
- Query Expression Syntax for Standard Query Operators (C#)
- LINQ to Objects (C#)

Recommended content

Aggregation operations (C#)

Learn about methods for performing an aggregation operation. An aggregation operation computes a single value from a collection of values.

into - C# Reference

into - C# Reference

Join Operations (C#)

A join of two data sources associates objects with objects that share an attribute across data sources. Learn about join methods in the LINQ framework in C#.

C# Features That Support LINQ

Learn about C# features to use with LINQ queries and in other contexts. These language constructs were introduced in C# 3.0.

Query Expression Syntax for Standard Query Operators (C#)

Learn about query expression syntax for standard query operators. See a list of standard query operators with equivalent query expression clauses.

Standard Query Operators Overview (C#)

The LINQ standard query operators provide query capabilities including filtering, projection, aggregation, and sorting in C#.

Quantifier Operations (C#) - LINQ

Learn about quantifier operations in LINQ. These methods, 'All', 'Any', and 'Contains', return a Boolean value indicating whether some or all elements in a sequence satisfy a condition.

Store the results of a query in memory

How to store results.

Show more