跟踪行为决定了 Entity Framework Core 是否将有关实体实例的信息保留在其更改跟踪器中。 **如果已跟踪某个实体，则该实体中检测到的任何更改都会在 `SaveChanges()` 期间永久保存到数据库。** EF Core 还将修复跟踪查询结果中的实体与更改跟踪器中的实体之间的导航属性（在EF中，外键被称为导航属性）。

备注

从不跟踪无键实体类型。 无论在何处提到实体类型，它都是指定义了键的实体类型。

提示

可在 GitHub 上查看此文章的示例。

跟踪查询

默认情况下，跟踪返回实体类型的查询。 这表示可以更改这些实体实例，然后通过 `SaveChanges()` 持久化这些更改。 在以下示例中，将检测到对博客分级所做的更改，并在 `SaveChanges()` 期间将这些更改永久保存到数据库中。

C#

```
var blog = context.Blogs.SingleOrDefault(b => b.BlogId == 1); blog.Rating = 5; context.SaveChanges();
```

在跟踪查询中返回结果时，EF Core 将检查上下文中是否已存在实体。 如果 EF Core 找到现有的实体，则返回同样的实例。 EF Core 不会用数据库值覆盖该实体中实体属性的当前值和原始值。 如果未在上下文中找到该实体，EF Core 将创建新的实体实例，并将其附加到上下文。 查询结果不会包含任何已添加到上下文但尚未保存到数据库中的实体。

非跟踪查询

在只读方案中使用结果时，非跟踪查询十分有用。 可以更快速地执行非跟踪查询，因为无需设置更改跟踪信息。 如果不需要更新从数据库中检索到的实体，则应使用非跟踪查询。 可以将单个查询替换为非跟踪查询。 非跟踪查询也会根据数据库中的内容提供结果，但不考虑本地更改或已添加的实体。

C#

```
var blogs = context.Blogs .AsNoTracking() .ToList();
```

还可以在上下文实例级别更改默认跟踪行为：

C#

```
context.ChangeTracker.QueryTrackingBehavior = QueryTrackingBehavior.NoTracking; var blogs = context.Blogs.ToList();
```

标识解析

由于跟踪查询使用更改跟踪器，因此 EF Core 将在跟踪查询中执行标识解析。 当具体化实体时，**如果 EF Core 已被跟踪，则会从更改跟踪器返回相同的实体实例。** 如果结果中多次包含相同的实体，则每次会返回相同的实例。 非跟踪查询不会使用更改跟踪器，也不会执行标识解析。 因此会返回实体的新实例（每次都会返回新的实例），即使结果中多次包含相同的实体也是如此。 此行为与 EF Core 3.0 之前的版本中的行为有所不同，请参阅早期版本。

从 EF Core 5.0 开始，可以在同一个查询中结合使用上述两种行为。 也就是说，可以使用非跟踪查询并对结果执行标识解析。 我们添加了另一个运算符 `AsNoTrackingWithIdentityResolution()`，就像添加 `AsNoTracking()` 可查询运算符一样。 QueryTrackingBehavior 枚举中也添加了一个关联项。 如果将查询配置为使用标识解析和非跟踪行为，生成查询结果时我们将在后台使用独立的更改追踪器，以便仅将每个实例具体化一次。 此更改追踪器不同于上下文中的更改追踪器，因此上下文不会追踪这些结果。 完全枚举查询后，该更改追踪器将超出范围，并根据需要对其进行垃圾回收。

C#

```
var blogs = context.Blogs .AsNoTrackingWithIdentityResolution() .ToList();
```

跟踪和自定义投影

即使查询的结果类型不是实体类型，默认情况下 EF Core 也会跟踪结果中包含的实体类型。 在以下返回匿名类型的查询中，会跟踪结果集中 `Blog` 的实例。

C#

```
var blog = context.Blogs .Select( b => new { Blog = b, PostCount = b.Posts.Count() });
```

如果结果集包含来自 LINQ 组合的实体类型，EF Core 将跟踪它们。

C#

```
var blog = context.Blogs .Select( b => new { Blog = b, Post = b.Posts.OrderBy(p => p.Rating).LastOrDefault() });
```

如果结果集不包含任何实体类型，则不会执行跟踪。 在以下查询中，我们返回匿名类型（具有实体中的某些值，但没有实际实体类型的实例）。 查询中没有任何被跟踪的实体。

C#

```
var blog = context.Blogs .Select( b => new { Id = b.BlogId, b.Url });
```

EF Core 支持执行顶级投影中的客户端评估。 如果 EF Core 具体化实体实例以进行客户端评估，则会跟踪该实体实例。 此处，由于我们要将 `blog` 实体传递到客户端方法 `StandardizeURL`，因此 EF Core 也会跟踪博客实例。

C#

```
var blogs = context.Blogs .OrderByDescending(blog => blog.Rating) .Select( blog => new { Id = blog.BlogId, Url = StandardizeUrl(blog) }) .ToList();
```

C#

```
public static string StandardizeUrl(Blog blog) { var url = blog.Url.ToLower(); if (!url.StartsWith("http://")) { url = string.Concat("http://", url); } return url; }
```

EF Core 不会跟踪结果中包含的无键实体实例（永远不会对_DbContext_中的更改进行跟踪，因此不会在数据库中插入、更新或删除这些更改）。 但 EF Core 会根据上述规则跟踪带有键的实体类型的所有其他实例。

在 EF Core 3.0 之前，某些上述规则的工作方式有所不同。 有关详细信息，请参阅早期版本。

旧版

在 3.0 版之前，EF Core 执行跟踪的方式有一些差异。 显著的差异如下：

- 如客户端与服务器评估页中所述，在 3.0 版之前，EF Core 支持在查询的任何部分中执行客户端评估。 客户端评估导致了实体的具体化，这不是结果的一部分。 因此 EF Core 分析了结果以检测要跟踪的内容。此设计有一些不同之处，如下所示：
  - 投影中的客户端评估（导致具体化，但未返回具体化的实体实例）未被跟踪。 以下示例未跟踪 `blog` 实体。

C#

```
var blogs = context.Blogs .OrderByDescending(blog => blog.Rating) .Select( blog => new { Id = blog.BlogId, Url = StandardizeUrl(blog) }) .ToList();
```

  - 在某些情况下，EF Core 未跟踪来自 LINQ 组合的对象。 以下示例未跟踪 `Post`。

C#

```
var blog = context.Blogs .Select( b => new { Blog = b, Post = b.Posts.OrderBy(p => p.Rating).LastOrDefault() });
```

- 只要查询结果中包含无键实体类型，整个查询就会进行非跟踪。 这表示不会跟踪结果中包含的带有键的实体类型。
- EF Core 曾经在非跟踪查询中执行标识解析。 它使用了弱引用来跟踪已返回的实体。 因此，如果结果集多次包含相同的实体，则每次会返回相同的实例。 尽管具有相同标识的上一个结果超出了范围并进行了垃圾回收，EF Core 也会返回新实例。