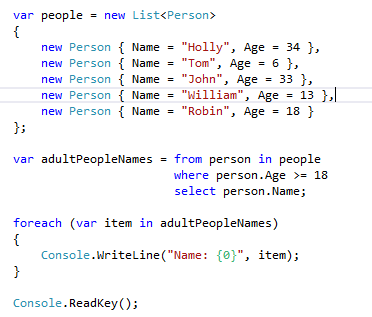


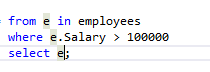


LINQ 2 Objects

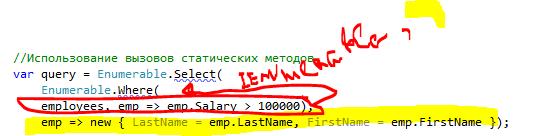
(информация храниться в памяти, а не удаленно)



одно и то же

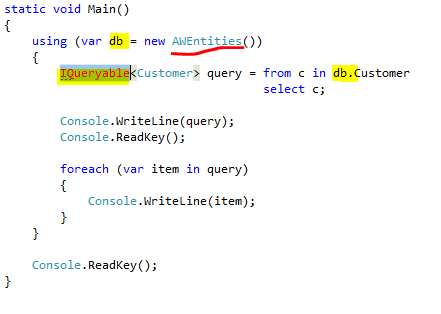


Или



IQueryable

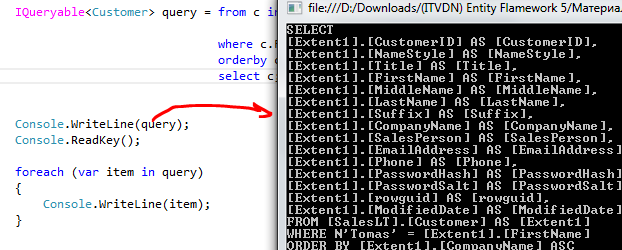
(во время выполнения запроса информация храниться удаленно, а не удаленно)



Но такой способ вернет IEnumerable

query = db.Customer;

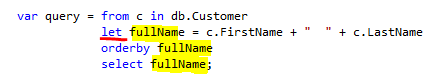
выводит запрос

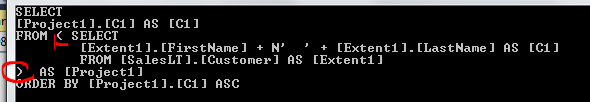


let

let - представляет новый локальный идентификатор, на который можно ссылаться в остальной части запроса.

Его можно представить, как локальную переменную видимую только внутри выражения запроса.





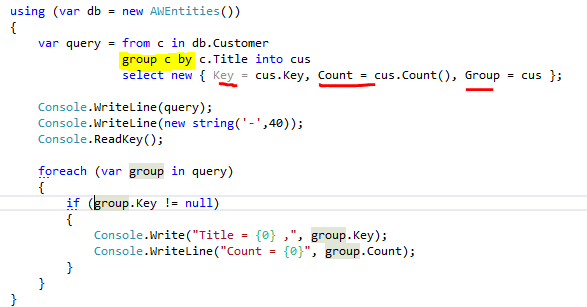
group c by c.Title into cus

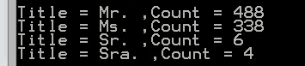
группировка в локальную таблицу cus

key - ключ

count - количество элементов в группировке

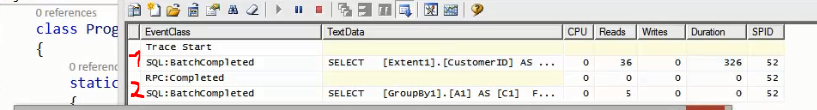
group - сами Customers

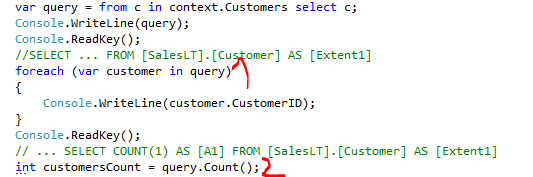




Удаленные запросы на сервере

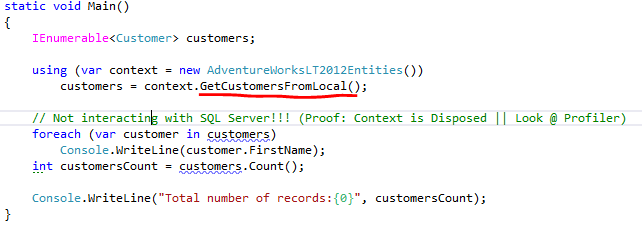
Оба раза был послан запрос серверу

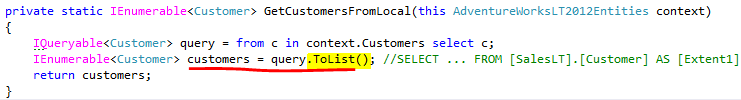




Данные запросы будут выполнены локально

Приведен к IEnumerable<Customer>

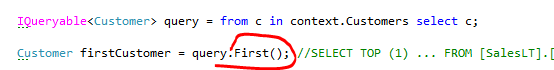




Обратился к серверу один раз когда приводил к List()

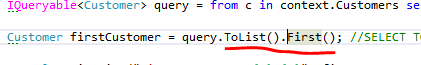


Чтение на сервере 2 раза загруженность CPU 0





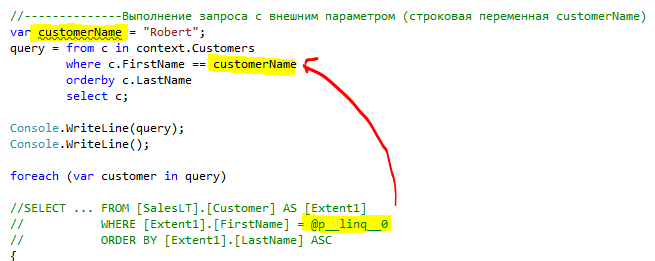
Чтение на сервере 36 раза загруженность CPU 36



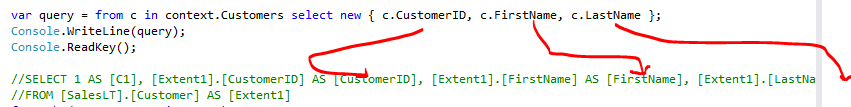


В случае обращения к серверу минимально тратиться оперативная память, но увеличивается нагрузка на сервер

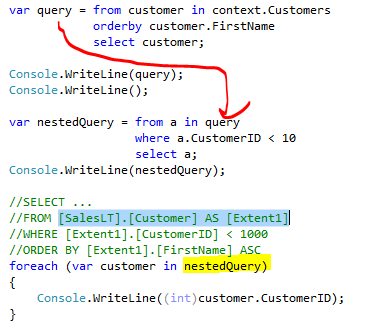
Когда имеется ссылка на локальную переменную, в данном случае используется SQL Parametr



При создании анонимного типа выбираются только нужные нам поля



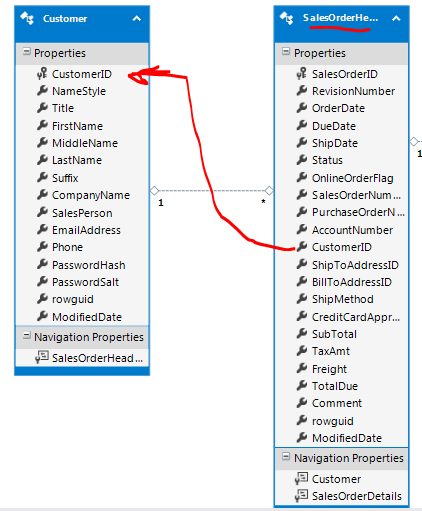
При вложенных запросах они будут скомбинированы в один и будут выполнены только когда мы обратимся для получения данных (в данном случае foreach)



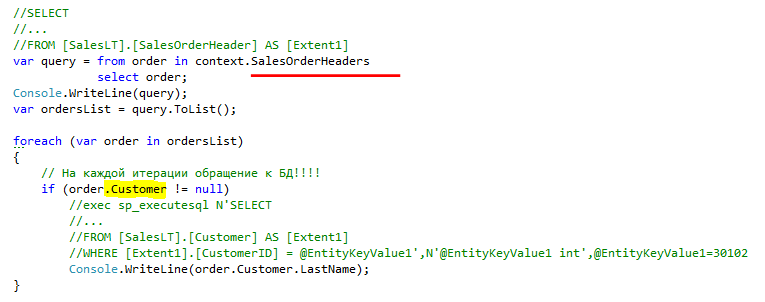
LazyLoad

Загружаются только те данные которые необходимо, при получении дополнительных будет создан запрос на сервер

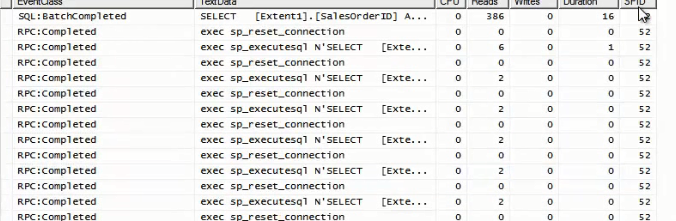
Имеется таблица заказы SalesOrderHeaders в которой внешний ключ Customer ссылающийся на сотрудников принявшие заказы



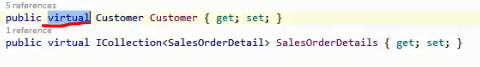
После первой выборки мы получим все записи каждого товара и при обращении к ключу (коллекции) Customer будет произведено обращение к другой таблице, и на каждой итерации будет создан запрос для обращения (записывается ссылка на специальный прокси объект)



Создано много запросов к серверу



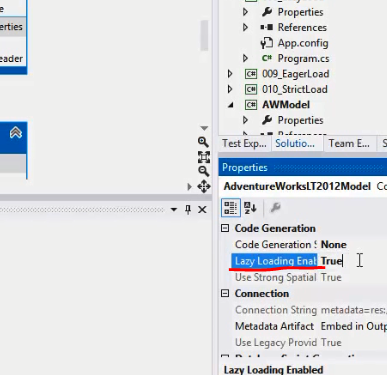
В классе заказы имеется виртуальное поле Customer и при создании объекта entity создает класс наследник в котором переопределяет данное свойство



context.Configuration.LazyLoadingEnabled = true; - изменение конфигурации из кода

Code-First defaults: true (по умолчанию ленивая загрузка включена)

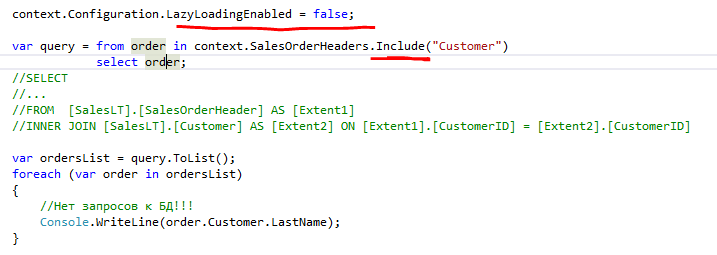
Если модель первая, то проверить



EagerLoad

Когда известно что будем обращаться к другим полям таблицы

context.SalesOrderHeaders.Include("Customer") – включи эту информацию во время выборки так как она мне будет нужна



StrictLoad

Явная загрузка

context.Customers.Load(); - загрузить в локальный контекст EntityFramework

обратиться к серверу два раза

