МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования   
«Санкт-Петербургский государственный технологический институт  
(технический университет)»  
СПбГТИ(ТУ)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.04.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного  проектирования и управления | | |
| Учебная дисциплина | |  | Основы разработки автоматизированных информационных систем | | |
| Курс | 1 | | | Группа | 499м |

Лабораторная работа 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тема: | Лабораторная работа 3 по курсу «Технологии разработки ПО» | | | | |
| Студент | |  |  |  | Плеханов А.А. |
| Преподаватели | |  |  |  | Корниенко И.Г., Федин К.А. |

Санкт-Петербург

2019

**Оглавление**

[1 Цель работы 3](#_Toc21902046)

[2 Постановка задачи 3](#_Toc21902047)

[3 Ход работы 3](#_Toc21902048)

[4 Разработка БД для программного комплекса OPTEL 3](#_Toc21902049)

[4.1 Выбор данных 3](#_Toc21902050)

[4.2 Добавление элемента 4](#_Toc21902051)

[4.3 Редактирование данных 5](#_Toc21902052)

[4.4 Удаление данных 6](#_Toc21902053)

[Выводы 7](#_Toc21902054)

**1 Цель работы**

Исследование базовых возможностей операционной системы, позволяющих повлиять на производительность разрабатываемого программного обеспечения.

**2 Постановка задачи**

1. Необходимо разработать код использующий технологии многопоточности, которые влияют и на подходы к проектированию архитектур разрабатываемого программного обеспечения.
2. Необходимо написать и проверить работоспособность запросов на языке SQL к разработанной базе данных. Запрос должны демонстрировать возможности и особенности следующих ключевых слов (SELECT, FROM, WHERE, UPDATE, DELETE, INNER JOIN, LEFT JOIN, RIGHT JOIN, GROUP BY, HAVING).

3 Многопоточность

Разберем многопоточность на примере программы, подбирающей пароли. Программа будет подбирать к выбранному логину пароли из файла, где их будет достаточно большое количество.

По сути – программа берет логин и пароль, отсылает их на web-сервер, чтобы проверить, ждет ответа, анализирует ответ. На то время, пока программа ждет ответа, процессор простаивает и следующий поток тоже может отправить запрос к серверу. Попробуем протестировать программу на 1 и, например, 10 потоках на скорость выполнения при всего 10 паролях.

Если в случае одного потока программа завершила работу за 1,46 сек, то в случае 10 потоков – за 0,27 сек, что, как видим, существенно быстрее.

4 Разработка БД для программного комплекса OPTEL

Для работы с базой данных необходимо иметь возможность: просматривать данные и редактировать их. Разберем этот функционал на примере таблицы заказов (для работы с другими таблицами все происходит практически идентично). Стоит заметить, что работа с БД будет происходить не SQL запросами, а на языке C# через Entity Framework, при эквивалентном функционале.

4.1 Выбор данных

Для просмотра данных необходимо выбрать их из БД, при этом, их надо отсортировать их по номеру заказа. Этот запрос выполняется командами SELECT и ORDER BY ASC, однако в Entity Framework возможно просто обратиться к необходимой таблице через контекст данных:

private OptelContext db = new OptelContext();

// GET: Orders

public async Task<ActionResult> Index(int? page)

{

var pageNumber = page ?? 1;

var pageContent = await db.Orders.OrderBy(i => i.OrderNumber).Include(i => i.FilmRecipe).ToPagedListAsync(pageNumber, Convert.ToInt32(WebConfigurationManager.AppSettings["ElementsPerIndexPage"]));

ViewBag.PageContent = pageContent;

return View();

}

Полученный список мы можем вывести пользователю (вывод отображен на рисунке 1).

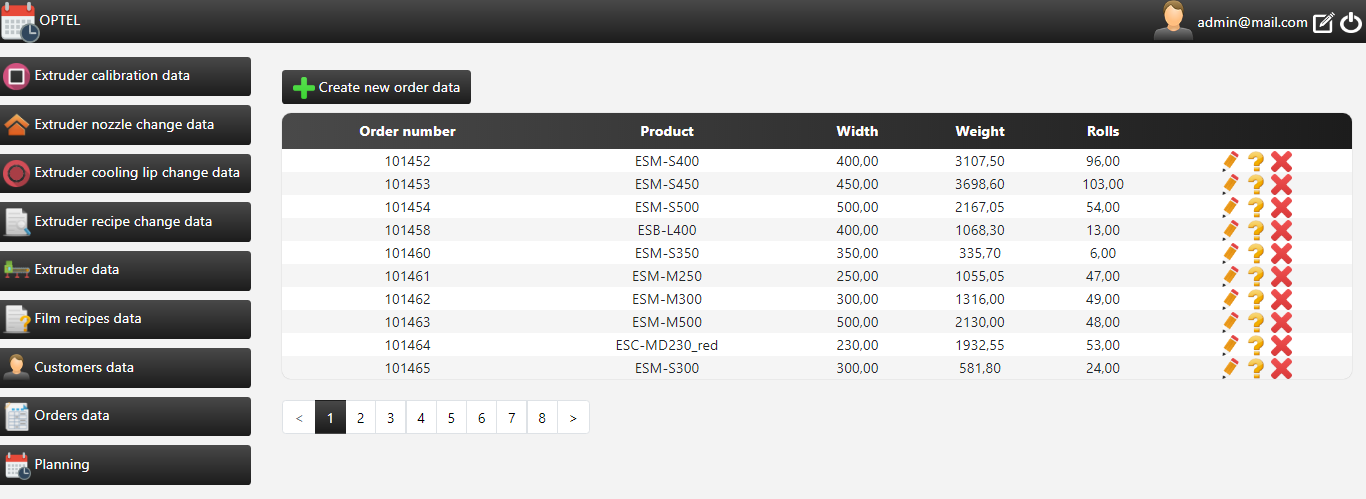


Рисунок 1 – Вывод отсортированных данных из БД пользователю

4.2 Добавление элемента

В программном комплексе OPTEL добавление элемента будет производиться через отдельную форму. В SQL для этого используется команда INSERT, посмотрим, как это можно реализовать на C#:

public async Task<ActionResult> Create([Bind(Include = "Id,OrderNumber,Product,Width,SetupTime,ProductionTime,ProductionInterruptionTime,TotalTime,FinishedGoods,Granules,Waste,QuanityInRunningMeter,RollWeightNet,Rolls,FilmRecipeId")] Order order)

{

if (ModelState.IsValid)

{

order.Id = Guid.NewGuid();

db.Orders.Add(order);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(order);

}

Добавим заказ с номером 101450 (чтобы он был самым первым) и посмотрим, выведется ли он на экран пользователю, что будет свидетельствовать об успешном сохранении заказа в БД (см. рисунок 2).

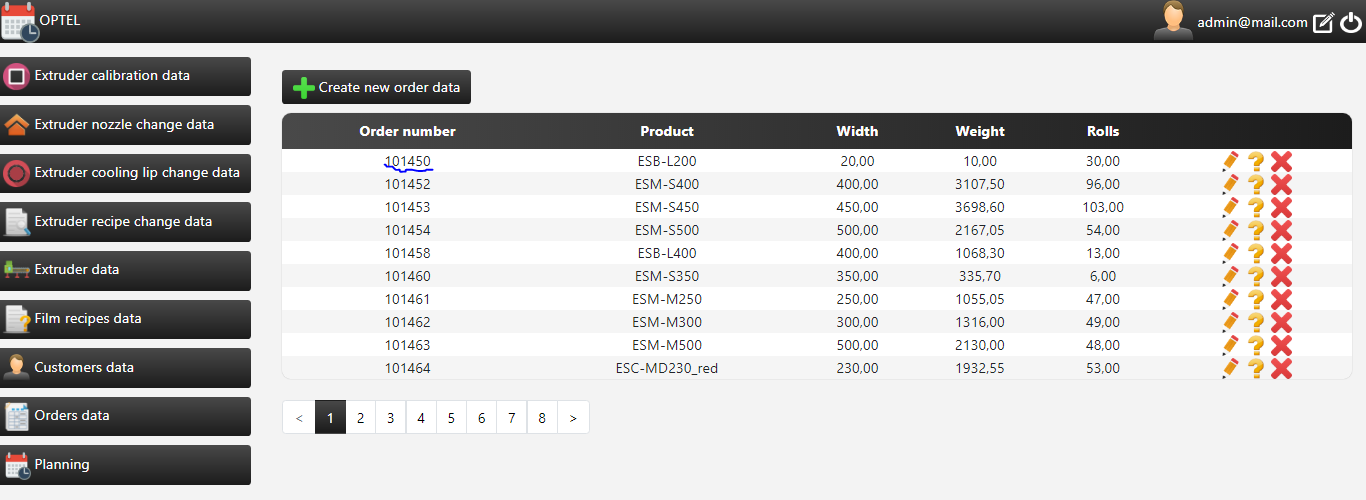


Рисунок 2 – Добавленный заказ в списке всех заказов

4.3 Редактирование данных

Изменение заказа так же проводится в отдельной форме. В SQL для редактирования данных в таблице используется команда UPDATE. Посмотрим на реализацию в C#:

public async Task<ActionResult> Edit([Bind(Include = "Id,OrderNumber,Product,Width,SetupTime,ProductionTime,ProductionInterruptionTime,TotalTime,FinishedGoods,Granules,Waste,QuanityInRunningMeter,RollWeightNet,Rolls,FilmRecipeId")] Order order)

{

if (ModelState.IsValid)

{

db.Entry(order).State = EntityState.Modified;

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(order);

}

Отредактируем наш заказ, изменив в нем данные о типе пленки и ширине и посмотрим, как программа выведет его пользователю (рисунок 3).

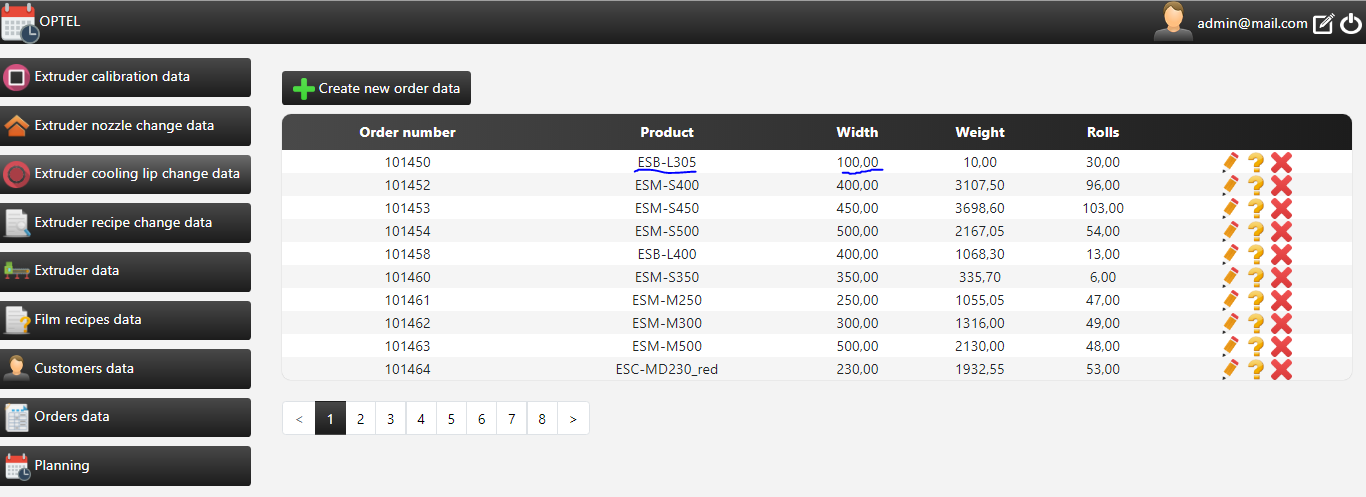


Рисунок 3 – Отображение данных заказа после операции изменения

4.4 Удаление данных

Для удаления данных в SQL используется команда DELETE. В C#, через Entity Framwork, это реализуется через контекст данных:

public async Task<ActionResult> DeleteConfirmed(Guid id)

{

Order order = await db.Orders.FindAsync(id);

db.Orders.Remove(order);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

Давайте теперь удалим созданный нами заказ и посмотрим, что выведет БД (рисунок 4). Нашего заказа больше нет в списке, а значит удаление прошло успешно.

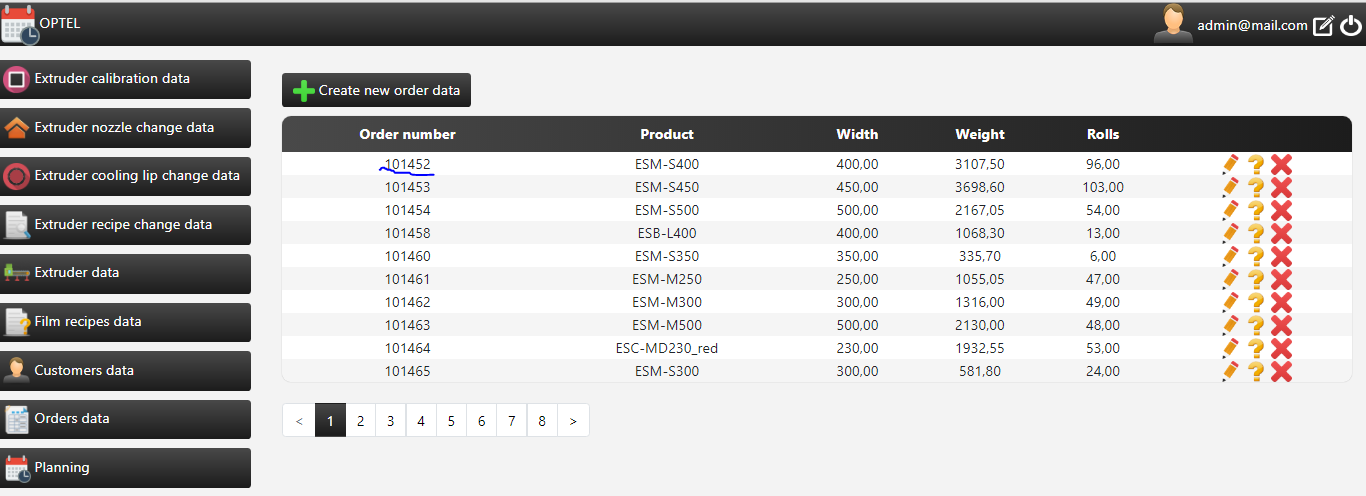


Рисунок 4 – Список заказов после операции удаления

Выводы

В результате распараллеливания задачи подбора пароля (с 1 потока до 10) было установлено, что скорость работы возросла почти в 7 раз. В итоге, можно сказать уверенно, что в общем случае многопоточность увеличивает скорость выполнения программы.

В блоке разработки БД были составлены запросы к БД для вывода, добавления, изменения и удаления данных через Entity Framwork. Результат работы программы показал, что все объявленные операции с данными прошли успешно.