Оптимизация процесса генерации отчетов

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Поздеев С.Н.

Осипович Т.А. – канд. эконом. наук, доцент

Цель работы – оптимизация процесса процесса генерации отчета в информационной системе планирования и прогнозирования бюджета.

Объект разработки – информационная система планирования и прогнозирования бюджет.

Предмет исследования – организация и взаимодействия данных.

Для оптимизации процесса генерации отчета решены следующие задачи: уменьшено количество обращений к базе данных (БД); повышена скорость чтения данных из БД.

Для уменьшения количества обращений к БД применено следующее решение: общие значения для всех пользовательских данных кэшированы. В случаи изменения данных - перезаписывается кэш. Данный кэш доступен всем пользователям, имеющим к ним доступ. Хранение кэша на сервере без использования не превышает 15 минут.

Для повышения скорости чтения данных использованы динамически генерируемые запросы, считывающие из БД только необходимые для создания отчета данные; решена задача по чтению данных со связью, представленной на рисунке 1.

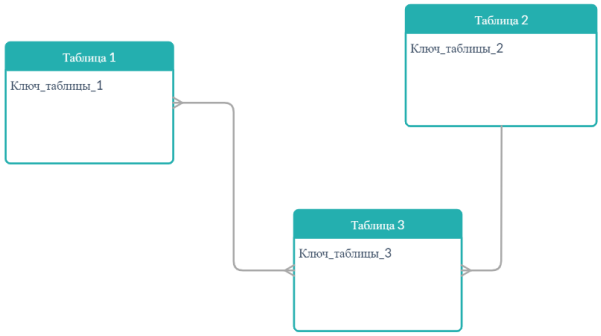

Рис.1 – Взаимодействие таблиц

Для использования динамически генерируемых запросов, необходимо отказать от использования framework для работы с БД и реализовать интерфейсы, при использовании языка программирования C#: IDbCommand – представляет инструкцию SQL, выполняемую при подключении к источнику данных, которую реализуют поставщики данных платформы .Net Core, имеющие доступ к реляционным базам данных; IDbConnection – представляет открытое подключение к источнику данных и реализуется поставщиками данных платформы .Net Core, которые имеют доступ к реляционным базам данных; IDataReader – предоставляет средства чтения одного или нескольких прямонаправленных потоков наборов результатов, полученных вследствие выполнения команды в источнике данных. Реализацию осуществляют поставщики данных .Net Core, которые имеют доступ к реляционным базам данных.

Использование данных интерфейсов для чтения записей из БД повысит скорость от 20% до 35% в сравнении с использованием Entity Framework (при чтении 100000 записей) и уменьшит объем хранимых данных на сервере, объем данных зависит от количества параметров необходимых для отчета.

Для увеличения производительности необходимо реализовать класс, наследуемый от NinjectModule. NinjectModule – это инструмент, используемый для регистрации различных типов в контейнеров IoC. Преимущество заключается в том, что эти модули хранятся в своих собственных классах. Это позволяет размещать разные уровни / сервисы в своих собственных модулях, то есть данное решение позволит сохранять в сессии клиента сформированные форматы данных для отображения числовых и строковых данных, так же настроенные форматы доступа к данным (доступ в зависимости от прав пользователя).

Список использованных источников:

1. Электронный ресурс – https://docs.microsoft.com/ru-ru/dotnet/api/system.data.idbconnection?view=netcore-3.1
2. Электронный ресурс – https://docs.microsoft.com/ru-ru/dotnet/api/system.data.idbcommand?view=netcore-3.1
3. П. Дейтел, Х. Дейтел, А. Уолд. Android для разработчиков. 3-е изд. — СПб.: Питер, 2016. — 512 с.: ил. — (Серия «Библиотека программиста»).