Для проверки сложности CAPTCHA были разработаны 7 моделей сверточнойсети. Каждая модель включает связки сверточных слоев, слоев пуллинга и фильтрационных матриц. Использование данных слоев в определенной комбинации помогает добиться максимальной эффективности сверточной сети. Комбинации порядка слоев влияют на карты признаков, генерируемые сетью.

На рисунке 1 представлен пример распознавания изображения одной из полученных моделью.

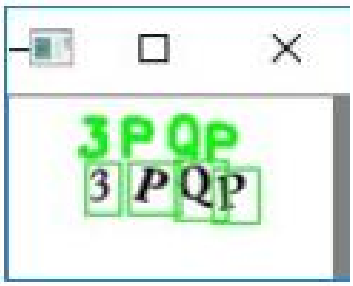

Рисунок 1 – Пример распознавания капчи

Представленная модель сверточной сети распознает 95.62% изображений плагина ReallySimpleCAPTCHA. Это говорит о том, что такой вариант CAPTCHA плохо защищают форму на сайте от спама.

УДК 004.43

ОПТИМИЗАЦИЯ ЗАПРОСОВ ПО ОБНОВЛЕНИЮ БОЛЬШИХ ОБЪЁМОВ ДАННЫХ

Якимович С.В., Вольский А.М.

Научный руководитель – Ковалева И.Л., к.т.н., доцент

Запросы на обновление записей в таблицах базы данных могут выполняться продолжительное время, в этом случае возникает необходимость оптимизации выполнения таких запросов.

Для этого были проанализированы действия, происходящие в MySQL, операционной системе и на аппаратном уровне при обновлении данных в таблице из текстового файла, содержащей 125 миллионов записей.

Ключом к решению задачи оптимизации являлся тот факт, что данные обновляются в том же порядке, в котором они были записаны в файл. Это означает, что каждый запрос будет обновлять случайное место в таблице, что повлечёт за собой потерю времени на позиционирование головок диска, потерю кэша файловой системы и потерю кэша базы данных [1].

Также важным фактом является то, что обновление каждой строки в MySQL состоит из трёх этапов: вычитка значений, сравнение старого и нового значения, запись значения [2].На основании информации о количестве совпавших и обновившихся строк, выведенных в результате запроса, было определено, что в анализируемой таблице изменялось только 10% записей. Это означает, что около 90% времени MySQL затрачивает на вычитку значений, а не на обновление данных. Поэтому оптимизация была связана с вычиткой значений.

Для этого было выполнено сравнение скорости произвольного и последовательного доступа к clusteredindex с помощью следующего запроса:

SELECT * FROM user WHERE user_id IN ($values)

В качестве параметра values использовался массив из 10 тысяч элементов, значения которых были случайными для проверки произвольного доступа, и, начиная с некоего случайного смещения, для проверки последовательного доступа. Результаты тестов представлены в таблице 1.

На основе вышеприведенных данных был найден способ оптимизации данной задачи: перед вставкой необходимо выполнить сортировку данных по первичному ключу.

Информация о выполнении запроса в случае, когда в качестве носителя данных выступает твердотельный накопитель, а не жёсткий диск, представлена в таблице 2.

Исследования проводились при отключенном кэшировании.

Таблица 1 – Информация о выполнении запросов при произвольном и последовательном доступе для жёсткого диска

Номер теста	Произвольный доступ	Последовательный доступ
1	40730	138
2	38035	143
3	38189	147
4	37161	138
5	40874	170

Таблица 2 – Информация о выполнении запросов при произвольном и последовательном доступе для твердотельного накопителя

Номер теста	Произвольный доступ	Последовательный доступ
1	5532	120
2	6401	114
3	5804	103
4	6012	118
5	5964	117

На основании значений из таблицы 2 можно сделать вывод, что использование твердотельного накопителя даёт преимущество во времени выполнения запроса, но не отменяет необходимости оптимизации.

Описанный подход позволил увеличить скорость выполнения запроса. Стоит отметить что прирост скорости может зависеть от разных факторов, например от включенного кеширования.

Литература

1. MolinaroA. SQLCборник Рецептов. – O’reily, Moskow. – P. 94-98.

2. SchwartzB.MYSQL. Оптимизация Производительности. – USA: O’reily. – 2010. – P. 27-29, 101-107