**Базы данных**

**Отчет по лабораторной работе №9, Лешук Дмитрий ПОИБМС 7-2**

**Цель**: Изучить такие объекты базы данных, как индексы, их разновидность, применение, создание, изменение, характеристики

Индекс – это объект базы данных, позволяющий ускорить поиск в определенной таблице, так как при этом данные организуются в виде сбалансированного бинарного дерева поиска. Как и любой другой объект базы данных, индекс может быть создан с помощью оператора CREATE, модифицирован с помощью ALTER и удален с помощью оператора DROP. Для одной таблицы возможно построение нескольких индексов.

Индексы бывают кластеризованные, некластеризованные, уникальные, неуникальные и др.

Обычно кластеризованные индексы создаются автоматически при создании таблицы если в ней присутствует первичный ключ (ограничение PRIMARY KEY). С помощью системной процедуры SP\_HELPINDEX можно получить перечень индексов, связанных с заданной таблицей – рисунок 1.1.

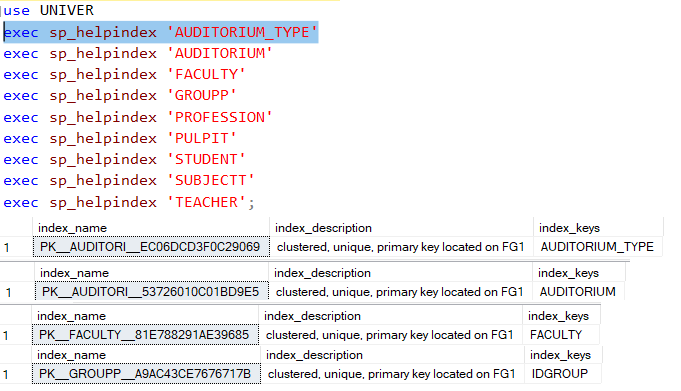


Рисунок 1.1 – Перечень индексов, связанных с таблицей

В первом задании необходимо было создать временную таблицу и заполнить её данными (не < 1000 строк). Далее анализируем стоимость выполнения запроса – 0.0066. Создадим кластеризованный индекс и выполним тот же самый запрос. Стоимость выполнения будет в 2 раза меньше – 0.0033, что показано на рисунке 1.2. Отметим также, что план выполнения запроса изменился : без индекса сначала проводилась сортировка, на которую отводилось 60% стоимости запроса, а потом уже «Scan Table». После создания индекса план запроса строится в одну операцию – поиск по кластеризованному индексу.

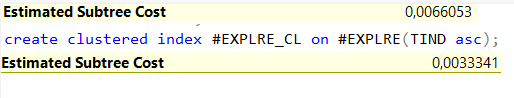


Рисунок 1.2 – Выполнение запроса с индексом и без

Следующее задание связано с некластеризованным индексом. Некластеризованные индексы не влияют на физический порядок строк в таблице. Снова создаем временную таблицу, заполняем её данными. Создаём составной индекс – индекс по нескольким столбцам. Этот индекс не применяется оптимизатором ни при фильтрации, ни при сортировке строк таблицы – Рисунок 1.3. Но, если хотя бы одно из индексируемых значений зафиксировать (задать одно значение), то оптимизатор применит индекс.

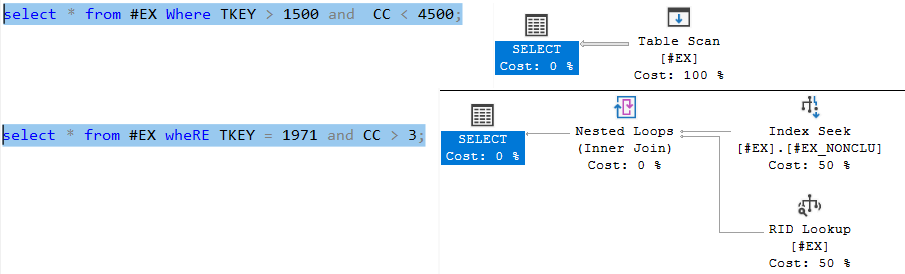


Рисунок 1.3 – Использование составного индекса

Некластеризованный индекс покрытия запроса позволяет включить в состав индексной строки значения одного или нескольких неиндексируемых столбцов. Используется с ключевым словом INCLUDE. Пример использования представлен на Рисунке 1.4.

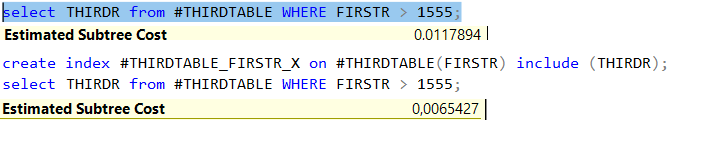


Рисунок 1.4 – Некластеризованный индекс покрытия

Следующий тип индекса – фильтруемый, представленный на рисунке 1.5. Может быть эффективным применение, если запросы основаны на WHERE-фильтрации строк.

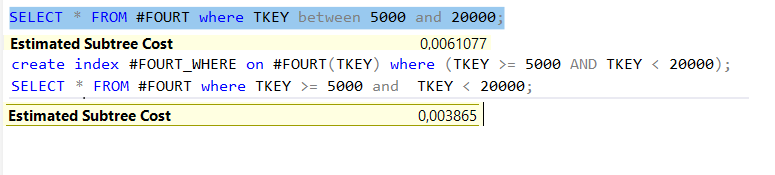


Рисунок 1.5 – Фильтруемый некластеризованный индекс

Далее была поставлена задача на оценку фрагментации индексов. Код запроса выглядит следующим образом – рисунок 1.6.

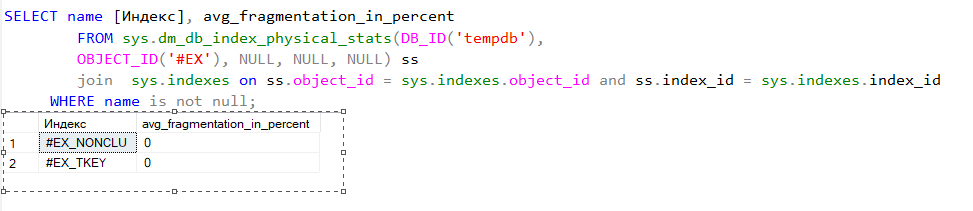


Рисунок 1.6 – Оценка фрагментации индексов

В таблице «#EX» мы создали два индекса, и в силу малого количества данных фрагментация равна нулю. Если с помощью конструкции INSERT-SELECT вставить в таблицу >100000 строк, то степень фрагментации значительно увеличится. При 128000 она приближается к 99% - Рисунок 1.7.



Рисунок 1.7 – Фрагментация ~97%

Для избавления от фрагментации индекса предусмотрены две специальные операции: реорганизация и перестройка индекса.

Реорганизация (REORGANIZE) выполняется быстро, но после нее фрагментация будет убрана только на самом нижнем уровне.

Операция перестройки (REBUILD) затрагивает все узлы дерева, поэтому после ее выполнения степень фрагментации равна нулю.

Используются эти операции следующим образом – рисунок 1.8.

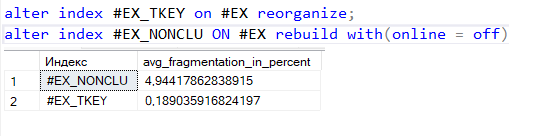


Рисунок 1.8 – Избавление от фрагментации индексов

Уровнем фрагментации можно в некоторой степени управлять, если при создании или изменении индекса использовать параметры FILLFACTOR и PAD\_INDEX. Параметр FILLFACTOR указывает процент заполнения индексных страниц нижнего уровня, указывается при создании таблицы с помощью «WITH (filfactor = N)». Использование приведено на рисунке 1.9.

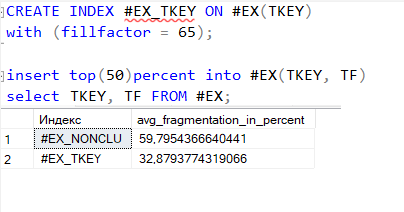


Рисунок 1.9 – Использование filfactor

**Вывод:**

В ходе данной лабораторной работы были изучены объекты базы данных – индексы. Были реализованы кластеризованные индексы и некластеризованные: покрытия, фильтруемый, составной для оптимизации запросов и сокращения затрат на выполнение. Рассмотрена степень фрагментации индексов и способы её сокращения – операторы REBUILD и REORGANIZE, а также управление степенью фрагментации при помощи filfactor.