Фильтрованные индексы SQL Server

Время чтения
3 мин.

Опубликовано
15.04.2025

В этом руководстве вы узнаете, как использовать отфильтрованные индексы SQL Server для создания оптимизированных некластеризованных индексов для таблиц.

Содержание

1. Введение в фильтрованные индексы SQL Server
2. Пример отфильтрованного индекса SQL Server
3. Преимущества отфильтрованных индексов

Введение в фильтрованные индексы SQL Server

Некластеризованный индекс, при правильном использовании, может значительно улучшить производительность запросов. Однако преимущества некластеризованных индексов имеют свою цену: хранение и обслуживание.

- Во-первых, для хранения копии данных ключевых столбцов индекса требуется дополнительное хранилище.
- Во-вторых, когда вы вставляете, обновляете или удаляете строки из таблицы, SQL Server должен обновить связанный некластеризованный индекс.

Было бы неэффективно, если бы приложения просто запрашивали часть строк таблицы. Вот почему в игру вступают отфильтрованные индексы.

Фильтрованный индекс — это некластеризованный индекс с предикатом, который позволяет указать, какие строки следует добавить в индекс.

Следующий синтаксис иллюстрирует, как создать отфильтрованный индекс:

```
1. CREATE INDEX index_name
2. ON table_name(column_list)
3. WHERE predicate;
```

В этом синтаксисе:

- Сначала укажите имя отфильтрованного индекса после предложения CREATE INDEX.
- Во-вторых, укажите имя таблицы со списком ключевых столбцов, которые будут включены в индекс.
- В-третьих, используйте предложение WHERE с предикатом, чтобы указать, какие строки таблицы следует включить в индекс.

Пример отфильтрованного индекса SQL Server

Для демонстрации мы будем использовать таблицу sales.customers из примера базы данных :

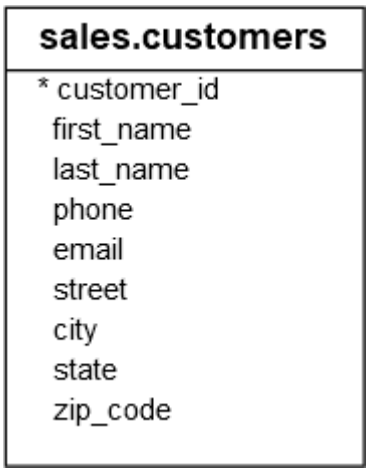


Таблица sales.customers содержит столбец phone, который содержит много значений NULL:

```
SELECT
    SUM(CASE
            WHEN phone IS NULL
            THEN 1
            ELSE 0
        END) AS [Has Phone],
    SUM(CASE
            WHEN phone IS NULL
            THEN 0
            ELSE 1
        END) AS [No Phone]
FROM
    sales.customers;
```

```
Has Phone   No Phone
----------- -----------
1267        178
(1 row affected)
```

Этот столбец с телефонами — хороший кандидат для отфильтрованного индекса.

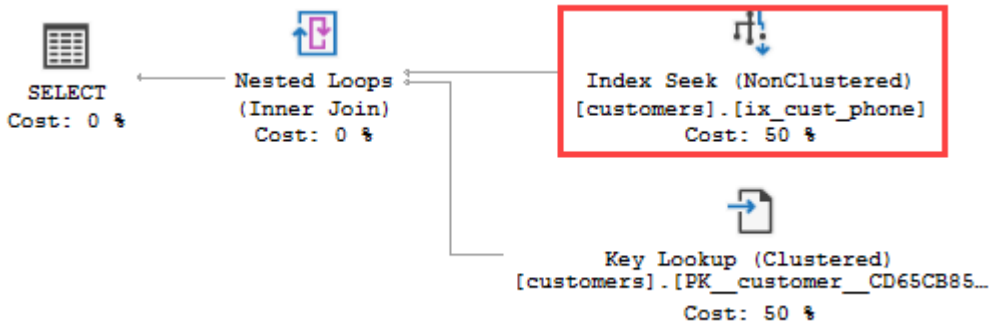
Этот оператор создает отфильтрованный индекс для столбца телефонов таблицы sales.customers:

```
CREATE INDEX ix_cust_phone
ON sales.customers(phone)
WHERE phone IS NOT NULL;
```

Следующий запрос находит клиента с номером телефона(281) 363-3309:

```
SELECT
    first_name,
    last_name,
    phone
FROM
    sales.customers
WHERE phone = '(281) 363-3309';
```

Вот предполагаемый план выполнения:



Оптимизатор запросов может использовать отфильтрованный индекс ix_cust_phone для поиска.

Обратите внимание, что для улучшения поиска ключей можно использовать индекс с включенными столбцами, который включает в себя столбцы first_name и last_name:

```
1. CREATE INDEX ix_cust_phone
2. ON sales.customers(phone)
3. INCLUDE(first_name, last_name)
4. WHERE phone IS NOT NULL;
```

Преимущества отфильтрованных индексов

Как упоминалось ранее, отфильтрованные индексы могут помочь вам сэкономить место, особенно когда ключевые столбцы индекса разрежены. Разреженные столбцы — это те, которые имеют много значений NULL.

Кроме того, отфильтрованные индексы снижают затраты на обслуживание, поскольку при изменении данных в связанной таблице необходимо обновлять только часть строк данных, а не все.

В этом руководстве вы узнали, как использовать отфильтрованные индексы SQL Server для создания оптимизированных некластеризованных индексов для таблиц.