Sprawozdanie – bazy danych

Kacper Andrzejewski 419925, IAD rok 2, grupa 1

Temat:

Analiza szczegółów wykonania zapytań związanych z kluczem głównym, operatorami logicznymi oraz indeksami.

Wstęp:

Klucze główne są powszechnie stosowanym narzędziem podczas pracy z bazami danych (SQL, PSQL itp.). Pozwalają na bardzo szybki dostęp do wielu danych z danej tabeli za pomocą np. przypisanym im ID. Dzięki temu wyszukiwanie, filtrowanie czy porównywanie danych przy pracy z dużymi zbiorami danych (duża ilość tabel) jest szybkie oraz skuteczne – dla użytkownika i dla programu. W tym sprawozdaniu zostanie przeprowadzona analiza kosztów operacji, czasu wykonania itd. przy wykonywaniu pewnego zapytania – w przypadku korzystania z klucza głównego oraz przeciwnym, zmiany operatora logicznego oraz działania indeksu. Wyniki i wnioski z nich wynikające umożliwią pokazanie, jak program zachowuje się w obu przypadkach dla każdej z rozpatrywanych sytuacji.

Wykonanie zapytań i analiza:

Zapytanie, które będzie wykorzystywane w tej części doświadczenia ma postać:

```
SELECT SalesOrderID, SalesOrderDetailID
FROM Sales.SalesOrderDetail
WHERE SalesOrderID = 43683 AND SalesOrderDetailID = 240;
```

Dotyczy ono wgranej do systemu uprzednio bazy „AdventureWorks2019”, która zawiera dużą ilość danych oraz jest dostępna za darmo. Statystyki, jakie będą porównywane to:

- Czas wykonania operacji
- Koszt operacji I/O
- Koszt CPU
- Ilość wierszy, która została wczytana do wykonania zapytania

Pierwszym krokiem jest wykonanie tego zapytania z użyciem narzędzia (dostępnego w MS SQL) – *Include Actual Execution Plan*. Umożliwia ono pokazanie statystyk dotyczących przeprowadzenia tej operacji. Dla istniejącego klucza głównego wyniosły one:

- Czas wykonania operacji: ~0.001s
- Koszt operacji I/O: ~0.0031
- Koszt CPU: ~0.0002
- Ilość wierszy (przetworzona): 1

Następnie, w celu dalszej analizy, należy usunąć klucz główny z tabel. Dokonuje się to poprzez wykonanie poniższego zapytania:

```
ALTER TABLE Sales.SalesOrderDetail
DROP CONSTRAINT PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID;
```

Po pomyślnym usunięciu klucza głównego należy ponownie wykonać poprzednią operację z identycznym zapytaniem (patrz. poprzednia strona) oraz odczytać dane statystyczne procesu. Wynoszą one:

- Czas wykonania: ~0.046s
- Koszt I/O: ~1.829
- Koszt CPU: ~1.336
- Ilość przetworzonych rzędów: 121317

Dodatkowo, należy dokonać analizy wyników po zmianie warunku „WHERE…AND…” na „WHERE…OR…”. W tym celu, należy przywrócić usunięty klucz główny za pomocą poniższego polecenia:

```
ALTER TABLE [Sales].[SalesOrderDetail] ADD CONSTRAINT
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] PRIMARY KEY CLUSTERED
(
[SalesOrderID] ASC,
[SalesOrderDetailID] ASC
)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
```

ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]

GO

EXEC sys.sp_addextendedproperty

@name=N'MS_Description',

@value=N'Primary key (clustered) constraint',

@level0type=N'SCHEMA', @level0name=N'Sales',

@level1type=N'TABLE', @level1name=N'SalesOrderDetail',

@level2type=N'CONSTRAINT',
@level2name=N'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID'

GO

Następnie należy wykonać polecenie, z wspomnianą wcześniej zmianą („AND” → „OR”). Daje ono następujące rezultaty:

- Czas wykonania: 0.028s
- Koszt I/O: ~0.206
- Koszt CPU: ~0.1336
- Ilość przetworzonych rzędów: 121317

Ostatnim krokiem w przeprowadzeniu analizy jest dokonanie zapytania ze słowem „WHERE” na dowolnej kolumnie nieposiadającej jeszcze indeksu. Należy je wykonać dla np. kolumny UnitPriceDiscount w tabeli Sales, w SalesOrderDetail. Ma ono postać:

*SELECT **

FROM Sales.SalesOrderDetail

WHERE UnitPriceDiscount = 0.15;

Wyniki przedstawiają się następująco:

- Czas wykonania: 0.053s
- Koszt I/O: ~0.9179
- Koszt CPU: ~0.1336
- Ilość przetworzonych rzędów: 121317

Kolejnym krokiem jest dodanie indeksu do tej kolumny przy użyciu:

```
CREATE NONCLUSTERED INDEX IX_SalesOrderDetail_UnitPrice_Include
ON Sales.SalesOrderDetail(UnitPriceDiscount);
```

Należy następnie ponowić poprzednie zapytanie. Po utworzeniu indeksu wyniki mają postać:

- Czas wykonania: 0.053s
- Koszt I/O: ~0.9179
- Koszt CPU: ~0.1336
- Ilość przetworzonych rzędów: 121317

Wnioski:

Dokonano analizy składającej się z 3 części. W pierwszej z nich porównano wyniki przed i po usunięciu klucza głównego z tabeli. Wyniki z obu testów wyszły różne – korzystniejsze dla istnienia klucza głównego (mniejszy koszt i krótszy czas wykonania polecenia w MSSQL). Oznacza to potwierdzenie tego, iż klucz główny pozwala na szybka identyfikację szukanych wierszy,

dopasowując je do siebie po odpowiednim Primary Key – np. ID. Proces jest wydajniejszy, ponieważ system błyskawicznie filtruje wprowadzone dane przy pomocy wspomnianego klucza, natomiast w przypadku jego usunięcia – musi dokonać przeszukiwania przez wszystkie wiersze w bazie i szukać dopasowania. W następnej części zadania dokonano pomiarów statystycznych dla polecenia „…WHERE…AND..," oraz „…WHERE…OR…". Korzystniejsze w obciążeniu i czasie okazały się wyniki dla operatora logicznego 'AND', ponieważ system szukał danych spełniających oba warunki, więc było ich mniej aniżeli podczas użycia 'OR', gdy wystarczy, aby jeden z warunków był spełniony. W konsekwencji operator koniunkcji wymaga mniejszego kosztu oraz czasu niż alternatywa. Finalnie, wyniki ostatniego zadania do wykonania implikują fakt, iż mimo dodania indeksu nieklastrowanego do wybranej kolumny, system automatycznie sam dostosowuje wybór indeksu (klastrowany lub nieklastrowany) – taki, który jest najbardziej optymalny przy wykonywaniu danego polecenia. W rezultacie, nieosiągalne było porównanie wyników dla obu rodzajów indeksów w bazie AdventureWorks2019.

źródła: https://learn.microsoft.com/pl-pl/sql/relational-databases/performance/display-an-actual-execution-plan?view=sql-server-ver16