309. Teoria

Widoki

Widok w bazie danych to zapytanie SELECT, które jest traktowane jak wirtualna tabela, którą można wykorzystać w innych zapytaniach. Widok jest tworzony na podstawie istniejących tabel lub innych widoków, a jego wynik jest przechowywany w pamięci i aktualizowany automatycznie, gdy zmieni się dane w podstawowych tabelach.

Wyróżnia się trzy rodzaje widoków:

```
Wirtualne: zwracają jedynie wybrany zestaw kolumn z jednej lub wielu tabel

Indeksowane: zapewniają dostęp do określonego zestawu danych w sposób podobny do indeksu (jako jedyne też trzymają dane)

Materializowane: tworzą fizyczne kopie danych na dysku, co zwiększa wydajność zapytań, ale wymaga dodatkowego miejsca na dysku i może wprowadzać opóźnienia w aktualizacji danych
```

Widoki różnią się od tabel tym, że nie przechowują danych w fizycznej formie, a jedynie definiują sposób ich pobierania i prezentacji. Widok może być traktowany jak zwykła tabela i wykorzystywany w zapytaniach SELECT, INSERT, UPDATE i DELETE, jednak jego wyniki są generowane dynamicznie na podstawie podstawowych tabel.

W skład składni tworzenia widoku wchodzi klauzula `CREATE VIEW`, po której następuje zapytanie SELECT, które określa sposób pobierania i formatowania danych. Widok może być aktualizowany, jeśli zapytanie SELECT spełnia określone kryteria, takie jak brak agregacji, brak grupowania po funkcji, brak wykorzystania `DISTINCT`, itp.

Przykładowa składnia tworzenia widoku wygląda następująco:

```
`CREATE VIEW [nazwa_widoku]
AS
    SELECT [kolumny]
    FROM [tabela]
    WHERE [warunki];`
```

Widok może zostać usunięty za pomocą klauzuli `DROP VIEW`, a jego definicja może zostać zmodyfikowana przez klauzulę `ALTER VIEW`.

Zalety widoków:

1. **Łatwość użytkowania** - Widoki umożliwiają wygodny dostęp do danych z różnych tabel i umożliwiają łączenie ich w sposób łatwy do zrozumienia, bez konieczności powtarzania zapytań lub operacji łączenia tabel w różnych miejscach.
2. **Bezpieczeństwo danych** - Widoki mogą służyć do ukrywania szczegółów przed użytkownikami, którzy nie mają uprawnień do ich przeglądania. Ograniczanie widoczności kolumn i wierszy pozwala kontrolować dostęp do danych i zapobiegać naruszeniom prywatności lub utracie poufnych informacji.
3. **Łatwe raportowanie** - Widoki są często wykorzystywane do tworzenia raportów i zestawień danych, co ułatwia pracę z danymi i przyspiesza procesy decyzyjne.

Procedury i triggery

Procedury w bazach danych to zbiory instrukcji SQL, które są grupowane razem i nazwane w celu łatwego wywoływania z innych zapytań lub aplikacji. Procedury są często wykorzystywane do wykonywania powtarzalnych czynności lub obliczeń na danych, co ułatwia programowanie, utrzymanie i zarządzanie bazami danych.

Składnia tworzenia procedury w języku SQL Server wygląda następująco:

```
CREATE PROCEDURE [nazwa_procedury]
[@parametr1 typ_danych, @parametr2 typ_danych, ...] AS [instrukcje SQL]
```

Parametry mogą być opcjonalne i służą do przekazywania danych do procedury. Instrukcje SQL są kodem, który definiuje operacje, jakie ma wykonać procedura. Po utworzeniu procedury można ją wywołać z innych zapytań lub aplikacji za pomocą instrukcji EXEC.

Procedury w SQL Serverze pozwalają na:

- Wykonanie określonych operacji na danych
- Optymalizację wykonywania zapytań poprzez unikanie powtarzającego się kodu
- Udostępnianie kodu pomiędzy wieloma aplikacjami
- Kontrolowanie dostępu do danych
- Ułatwienie zarządzania bazą danych i jej utrzymanie

Trigger w SQL Serverze to specjalna procedura, która jest automatycznie uruchamiana w odpowiedzi na określone zdarzenia w bazie danych, takie jak dodawanie, usuwanie lub aktualizacja wierszy w tabeli. Trigger może wykonywać różne operacje, takie jak aktualizacja danych, wywołanie procedury lub wstawienie danych do innej tabeli.

Składnia tworzenia triggera wygląda następująco:

```
CREATE TRIGGER [nazwa_triggera]
ON [nazwa_tabeli] AFTER INSERT, UPDATE, DELETE AS [instrukcje SQL]
```

Instrukcje SQL zawierają kod, który zostanie uruchomiony w odpowiedzi na określone zdarzenia w tabeli. Trigger może być wykorzystywany do automatyzacji procesów lub do śledzenia zmian w danych.

Zalety triggerów to:

- Automatyzacja procesów biznesowych
- Unikanie konieczności ręcznego wykonywania operacji na danych
- Śledzenie zmian w danych
- Ochrona przed błędami użytkownika lub nieuprawnionym dostępem do danych
- Kontrolowanie integralności danych w bazie danych.

Indeksy

Indeksy w bazach danych to struktury, które umożliwiają szybkie wyszukiwanie i sortowanie danych. Indeksy działają jak spis treści w książce, pozwalając na szybkie odnalezienie określonych wierszy w tabeli bez konieczności przeszukiwania całej tabeli.

Indeksy są tworzone na jednej lub kilku kolumnach tabeli i pozwalają na optymalizację wykonywania zapytań SELECT. Indeksy zmniejszają czas potrzebny na wykonanie zapytań, ale jednocześnie zwiększają miejsce zajmowane przez bazę danych i spowalniają czas wykonywania operacji INSERT, UPDATE i DELETE, ponieważ wymagają aktualizacji indeksów.

Jeżeli stworzymy tabele bez indeksu (np. bez klucza głównego, na którym automatycznie tworzony jest indeks) to przy łączeniu danych lub przy ograniczeniu WHERE to silnik musi przeskanować całą tabelę.

Typy indeksów, które można utworzyć w bazie danych to:

- Indeks podstawowy (ang. clustered index)
- Indeks niestandardowy (ang. nonclustered index)

Indeksy są wykorzystywane w celu optymalizacji wykonywania zapytań SELECT i umożliwiają szybsze wyszukiwanie i sortowanie danych w tabelach. Indeksy są szczególnie przydatne w tabelach, które zawierają duże ilości danych lub w tabelach, które są często wykorzystywane do wyszukiwania i sortowania danych.

Clustered index

Indeks podstawowy, znany również jako indeks klastrowany (ang. clustered index), to struktura indeksu, która sortuje dane fizycznie w tabeli na podstawie wartości kolumny klucza głównego. Oznacza to, że wiersze w tabeli są ułożone w kolejności alfabetycznej lub numerycznej, w zależności od typu danych, kolumny indeksowanej.

Indeks podstawowy wyróżnia się z pozostałych typów indeksów tym, że istnieje tylko jeden indeks podstawowy na jedną tabelę. Indeks podstawowy może mieć kolumnę indeksowaną lub wiele kolumn indeksowanych. W przypadku, gdy klucz główny jest złożony z kilku kolumn, indeks podstawowy jest tworzony na całym kluczu głównym.

Korzystanie z indeksu klastrowanego pozwala na szybki dostęp do posortowanych danych w tabeli, ponieważ system bazodanowy może odczytać tylko te wiersze, które znajdują się w określonym przedziale wartości indeksu. Dzięki temu indeks podstawowy jest szczególnie przydatny w przypadku, gdy tabela zawiera duże ilości danych, które należy szybko przeszukać.

Jednak korzystanie z indeksu podstawowego wiąże się z pewnymi wadami, ponieważ indeksowanie danych wymaga ich ponownego układania w pamięci, co może spowodować opóźnienia w czasie wykonania operacji INSERT, UPDATE i DELETE. Ponadto, gdy tabela jest zbyt duża, tworzenie indeksu podstawowego może wymagać znacznych zasobów dyskowych.

Ważne jest również, aby pamiętać, że zmiana klucza głównego w tabeli z indeksem klastrowanym może spowodować dużą liczbę przeprowadzonych operacji aktualizacji i odbudowania indeksu. Z tego powodu indeks podstawowy powinien być starannie przemyślany przed jego utworzeniem, a zmiana klucza głównego w tabeli z istniejącym indeksem podstawowym powinna być przeprowadzona z ostrożnością i po dokładnym przemyśleniu.

Struktura tworzona przez SQL Server to B-tree (lub balanced tree). Ażeby poprawnie zrozumieć jej działanie, poniżej znajduje się wizualizacja algorytmu:

https://www.cs.usfca.edu/~galles/visualization/BTree.html

Non-clustered index

Indeks niestandardowy (ang. nonclustered index) to struktura indeksu, która umożliwia szybsze wyszukiwanie danych w tabeli na podstawie wartości indeksowanej kolumny lub kolumn. Indeks niestandardowy działa jako struktura drzewiasta, w której liście zawierają wskaźniki do oryginalnych wierszy w tabeli, a każdy węzeł w drzewie zawiera wartości indeksowanej kolumny i wskaźnik do węzła podrzędnego.

W odróżnieniu od indeksu podstawowego, który sortuje dane fizycznie w tabeli, indeks niestandardowy nie zmienia kolejności wierszy w tabeli. Zamiast tego, indeks niestandardowy umożliwia szybkie przeszukiwanie danych w tabeli na podstawie wartości indeksowanej kolumny. Dzięki temu indeks niestandardowy jest szczególnie przydatny w przypadku, gdy tabela zawiera duże ilości danych, które należy przeszukiwać.

Indeks niestandardowy może zawierać kolumny indeksowane lub kolumny nieindeksowane, a każda tabela może mieć wiele indeksów niestandardowych. Indeks niestandardowy można tworzyć również na wartościach nieliczbowych, takich jak tekstowe lub daty.

Korzystanie z indeksu niestandardowego ma kilka zalet. Po pierwsze, indeks niestandardowy umożliwia szybkie wyszukiwanie danych w tabeli, co pozwala na szybsze wykonywanie zapytań SELECT. Po drugie, indeks niestandardowy pozwala na optymalizację wydajności bazy danych, ponieważ zmniejsza czas potrzebny na przeszukiwanie całej tabeli. Po trzecie, indeks niestandardowy umożliwia łatwe sortowanie danych na podstawie wartości indeksowanej kolumny.

Jednak korzystanie z indeksu niestandardowego wiąże się również z pewnymi wadami. Tworzenie indeksu niestandardowego wymaga dodatkowego miejsca w pamięci, co może spowodować spowolnienie wykonywania operacji INSERT, UPDATE i DELETE. Ponadto, indeksowanie kolumn nieliczbowych, takich jak tekstowe lub daty, może być bardziej kosztowne i zwiększać rozmiar indeksu.

Podsumowując, indeks niestandardowy umożliwia szybsze wyszukiwanie i sortowanie danych w tabeli na podstawie wartości indeksowanej kolumny. Indeks niestandardowy jest szczególnie przydatny w przypadku, gdy tabela zawiera duże ilości danych, które należy szybko przeszukać.

Inne ważne zagadnienia

DELETE, TRUNCATE i DROP są poleceniami SQL służącymi do usuwania danych lub całych tabel z bazy danych, jednak różnią się w sposób, w jaki działają i jakie skutki mają dla danych i struktury bazy danych.

DELETE - polecenie DELETE jest używane do usuwania wierszy z tabeli na podstawie określonych kryteriów. Usunięcie wierszy za pomocą polecenia DELETE powoduje, że dane są trwale usuwane z tabeli, ale struktura tabeli pozostaje nienaruszona. Wiersze usunięte za pomocą polecenia DELETE mogą zostać przywrócone przez wykorzystanie polecenia ROLLBACK.

TRUNCATE - polecenie TRUNCATE służy do usuwania wszystkich wierszy z tabeli. TRUNCATE jest szybsze od DELETE, ponieważ usuwa dane bez zachowywania dziennika transakcji. Oznacza to, że dane zostaną trwale usunięte z tabeli, ale nie będą mogły zostać przywrócone przez wykorzystanie polecenia ROLLBACK. TRUNCATE zachowuje strukturę tabeli, a po wykonaniu tego polecenia można dalej używać tej samej tabeli.

DROP - polecenie DROP służy do usuwania całej tabeli z bazy danych. DROP jest najbardziej radykalnym ze wszystkich poleceń i powoduje, że tabela zostanie całkowicie usunięta, wraz ze wszystkimi danymi i strukturą. Po wykonaniu polecenia DROP nie ma możliwości przywrócenia usuniętej tabeli ani jej danych.