A logo with text and hands

Description automatically generated with medium confidenceA blue and white logo

Description automatically generated

**Univerzitet u Nišu**

**Elektronski faktultet**

**Obrada upita kod MS SQL baze podataka**

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

Student: Profesor:

Katarina Maksimović, 1599 Prof. dr. Aleksandar Stanimirović

Niš, April 2024. godina

Sadržaj

[1. Uvod 3](#_Toc164515310)

[2. MSSQL Server 3](#_Toc164515311)

[2.1. Arhitektura MS SQL Servera 4](#_Toc164515312)

[2.2. SQL (Structured Query Language) u obradi upita 5](#_Toc164515313)

[3. Proces obrade upita 6](#_Toc164515314)

[3.1. Parsiranje SQL upita 7](#_Toc164515315)

[3.1.1. Ulazni tokeni 8](#_Toc164515316)

[3.1.2. Generisanje sintaksnog stabla (syntax tree) 8](#_Toc164515317)

[3.1.3. Sintaksna provera 9](#_Toc164515318)

[3.1.4. Semantička provera 10](#_Toc164515319)

[3.1.5. Shared Pool provera 11](#_Toc164515320)

[3.2. Optimizacija upita 12](#_Toc164515321)

[3.2.1. Generisanje različitih planova izvršavanja (Execution planova) 12](#_Toc164515322)

[3.2.2. Row Source Generation 14](#_Toc164515323)

[3.3. Execution plan 14](#_Toc164515324)

[4. Zaključak 20](#_Toc164515325)

[5. Literatura 21](#_Toc164515326)

# Uvod

Obrada upita predstavlja važan proces prilikom interakcije sa bazama podataka, kao što je MS SQL Server baza podataka. MS SQL Server je jedan od najmoćnijih sistema za upravljanje bazama podataka koji se široko koristi u različitim industrijama i poslovnim sektorima.

Obrada upita je proces koji omogućava korisnicima da pristupaju, filtriraju, modifikuju i analiziraju podatke u bazi podataka. Kroz obradu upita, korisnici mogu da izvrše različite operacije nad podacima, kao što su izvlačenje informacija putem SELECT upita, dodavanje novih podataka kroz INSERT upite, ažuriranje postojećih podataka putem UPDATE upita i brisanje nepotrebnih podataka kroz DELETE upite. Osim toga, obrada upita omogućava i kompleksnije operacije poput grupisanja podataka, spajanja više tabela, korišćenje funkcija i procedura, što je ključno za razvoj sofisticiranih aplikacija i izveštavanje. Stoga, razumevanje procesa obrade upita je od presudnog značaja za efikasno korišćenje MS SQL Servera i ostvarivanje optimalnih performansi baze podataka.

# MSSQL Server

Microsoft SQL Server je moćan sistem za upravljanje relacionim bazama podataka koji se koristi za skladištenje, manipulaciju i upravljanje podacima u različitim poslovnim okruženjima. Razvijen je od strane Microsoft-a 1989. godine kao SQL Server 1.0 pisan u C/C++.

SQL Server nudi različite verzije, od besplatnih verzija poput SQL Server Express Edition-a do naprednih verzija kao što su SQL Server Standard i SQL Server Enterprise Edition. Svaka verzija ima svoje karakteristike i ograničenja, što omogućava korisnicima da odaberu odgovarajuću verziju u skladu sa potrebama i zahtevima njihovih projekata.

SQL Server podržava rad na više platformi, uključujući Windows i Linux. To omogućava veću fleksibilnost u izboru operativnog sistema u zavisnosti od specifičnih potreba korisnika i infrastrukture.

Jedna od ključnih karakteristika SQL Servera je njegova skalabilnost. SQL Server omogućava horizontalno i vertikalno skaliranje, što znači da se može prilagoditi rastućem broju korisnika i većim količinama podataka.

SQL Server takođe nudi bogat ekosistem alata i tehnologija koji olakšavaju razvoj, upravljanje i analizu podataka. To uključuje alate za upravljanje bazama podataka poput SQL Server Management Studio (SSMS), alate za integraciju podataka poput SQL Server Integration Services (SSIS), alate za analizu podataka poput SQL Server Analysis Services (SSAS) i alate za izveštavanje poput SQL Server Reporting Services (SSRS).

# Arhitektura MS SQL Servera

Microsoft SQL Server je klijent-server arhitektura. MS SQL proces počinje kada klijentska aplikacija pošalje zahtev. SQL Server prihvata, procesira i odgovara na zahtev sa obrađenim podacima. SQL Server se sastoji od dve glavne komponenta:

* Database Engine
* SQLOS

Database Engine je osnovna komponenta arhitekture SQL Servera koja služi za skladištenje, obradu i bezbednost podataka. SQL Server podržava najviše 50 instanci baze podataka na jednom računaru. Pruža kontrolisan pristup i brzu obradu transakcija kako bi zadovoljio zahteve velikih aplikacija. Čak i objekti baze podataka poput stored procedura, view-a i trigger-a se takođe kreiraju i izvršavaju putem Datebase Engine-a. Sastoji se od Relational Engine-a za obradu upita i Storage Engine-a koji upravlja fajlovima baze podataka, indeksima itd.

Relational Engine: Glavni funkcija Relational Engine-a je obrada upita, ali ima niz drugih funkcionalnosti. Obično zahteva podatke od Storage Engine-a za dati ulazni upit i procesira izlaz na osnovu toga. Postoje 3 glavne komponente Relational Engine-a:

* Parser komandi – Prva komponenta Relational Engine-a koja prima podatke upita. Uglavnom proverava sintaksne i semantičke greške upita. I na kraju generiše stablo upita.
* Optimizator – Glavni zadatak optimizatora je pronalaženje najjeftinijeg, a ne najboljeg, ekonomičnog plana izvršavanja. I optimizacija se uglavnom vrši za DML (SELECT, INSERT, UPDATE, DELETE) naredbe, a ne za sve upite. Krajnji cilj je minimizacija vremena izvršavanja upita.
* Izvršilac upita – Poziva Access Method. Pruža plan izvršenja za logiku dobijanja podataka potrebnu za izvršenje. Nakon što podaci stignu iz Storage Engine-a, rezultat ide u Protocol layer. I konačno, podaci se šalju korisniku.

Storage Engine: Odgovoran je za skladištenje i povlačenje podataka sa sistemima za skladištenje poput diskova i SAN-a. Ima sledeće tri glavne komponente:

* Access Method – Određuje da li upit jeste ili nije SELECT naredba. I zatim poziva Buffer i Transfer Manager na osnovu toga.
* Buffer Manager – Upravlja osnovnim funkcijama za keširanje plana(Plan Cache), Data Parsing-a i Dirty Page-a.
* Transaction Manager – Upravlja Non-SELECT transakcijama uz pomoć Log i Lock Managers-a.

SQLOS (The SQL Server Operating System) je poseban sloj aplikacije na najnižem nivou SQL Server Database Engine-a koji pokreću i SQL Server i SQL Reporting Services. Predstavljen je u SQL Serveru 2005 godine. Nalazi se ispod Relational Engine-a i Storage Engine-a. Pruža operativne sistemske usluge kao što su upravljanje memorijom i I/O, uključujući i druge usluge koje uključuju obradu izuzetaka i sinhronizaciju. Zaprao vrši sledeće ključne funkcije za SQL Server:

* Planiranje i završetak IO-a: SQLOS je odgovoran za signalizacione niti kada je IO završen.
* SQLOS je odgovoran za upravljanje sinhronizacijom niti.
* Framewok za obradu izuzetaka.
* Detekcija i upravljanje deadlock-om.
* SQLOS može kontrolisati koliko memorije komponenta unutar SQL Servera koristi.
* Hosting usluge za spoljne komponente kao što su CLR i MDAC.

Dakle, može se reći da ima odlične sposobnosti upravljanja resursima i uvek se brine da SQL Server radi.

# SQL (Structured Query Language) u obradi upita

Structured Query Language (SQL) je standardni jezik koji se koristi za skladištenje i obradu podataka u relacionim bazama podataka, uključujući i Microsoft SQL Server. Razvijen je od strane IBM-a ranih 1970-ih i postao je komercijalno dostupan 1979. godine. SQL omogućava korisnicima da definišu, manipulišu i upravljaju podacima kroz različite vrste upita.

SQL omogućava izvođenje različitih operacija nad podacima, uključujući izvlačenje podataka (SELECT), dodavanje novih podataka (INSERT), ažuriranje postojećih podataka (UPDATE) i brisanje nepotrebnih podataka (DELETE).

Takođe, SQL omogućava i kompleksne operacije poput filtriranja, sortiranja, grupisanja i spajanja podataka iz različitih tabela, što ga čini moćnim alatom za analizu i manipulaciju podacima.

SQL (Structured Query Language) je značajan iz nekoliko razloga:

* Predstavlja standard za interakciju sa sistemima za upravljanje relacionim bazama podataka (RDBMS).
* Omogućava korisnicima da brzo izvuku određene podskupove podataka iz ogromnih baza podataka. Pruža komande za dodavanje, ažuriranje, brisanje i modifikaciju podataka, olakšavajući besprekorno manipulisanje podacima.
* Služi kao industrijski standardni jezik za operacije nad relacionim bazama podataka. Njegova standardizovana sintaksa i komande osiguravaju doslednost i kompatibilnost među različitim sistemima baza podataka, promovišući interoperabilnost.
* Pruža robustne mehanizme za obezbeđivanje integriteta podataka, primenu ograničenja, definisanje veza između tabela i implementaciju sigurnosnih mera poput autentifikacije korisnika i kontrole pristupa.
* Podržava skalabilne arhitekture baza podataka, omogućavajući organizacijama efikasno upravljanje rastućim količinama podataka. Pored toga, SQL omogućava administratorima baza podataka optimizaciju performansi upita putem indeksiranja, optimizacije upita i drugih tehnika.
* Olakšava naprednu analizu podataka omogućavajući korisnicima da formulišu kompleksne upite, agregiraju podatke, generišu izveštaje i izvlače uvide iz baza podataka. Igra ključnu ulogu u poslovnoj inteligenciji, analitici podataka i procesima donošenja odluka.
* Besprekorno se integriše sa različitim programskim jezicima i framework-ovima. Programeri koriste SQL da ugrade operacije baza podataka u aplikacije, veb sajtove i softverska rešenja, osiguravajući postojanost podataka, pouzdanost i funkcionalnost.
* Korišćenjem SQL transakcija, organizacije mogu održavati doslednost i integritet podataka tokom višestrukih istovremenih operacija nad bazom podataka, osiguravajući da se promene ili potpuno izvrše ili ponište u slučaju neuspeha.
* Pomaže arhitektima i dizajnerima baza podataka u modeliranju složenih struktura podataka, definisanju veza, uspostavljanju pravila normalizacije i dizajniranju efikasnih šema baza podataka koje se slažu sa zahtevima organizacije.

# Proces obrade upita

Obrada upita u SQL Serveru se vrši pomoću Relational Engine-a. To je proces uzimanja T-SQL naredbni, slanje zahteva Storage Engine-u i vraćanje potrebnih rezultata.

SQL Server preduzima tri koraka za obradu upita: parsiranje, optimizaciju i izvršenje. Ovi koraci su prikazani na slici 1. Prva dva koraka izvodi Relational Engine. Rezultat drugog koraka je optimizovani plan koji je zakazan i tokom kojeg se upućuju pozivi Storage Engine-u da bi se preuzeli podaci koji postaju rezultati upita koji se izvršava.

A diagram of a software company

Description automatically generated with medium confidence

Slika 1. Dijagram obrade upita kod MS SQL baze podataka

Na slici 2 je prikazan detaljniji dijagram ovih koraka.

A diagram of a software process

Description automatically generated

Slika 2. Detaljniji dijagram obrade upita kod MS SQL baze podataka

# Parsiranje SQL upita

Parsiranje SQL upita je prvi korak u procesu obrade upita. Tokom ovog koraka, SQL Server prima SQL upit u tekstualnom obliku i analizira ga kako bi razumeo njegovu strukturu i semantiku. Ovaj proces uključuje identifikaciju ključnih elemenata upita kao što su ključne reči (npr. SELECT, FROM, WHERE), imena tabela, kolona i uslova. SQL Server takođe proverava ispravnost sintakse upita, identifikuje eventualne greške i priprema upit za dalju obradu. Vrši proveru i Shared Pool-a. Kako svaki upit poseduje heš kod tokom svog izvršavanja, ovim se ispituje da li taj heš kod već postoji u Shared Pool-u. Ako kod postoji u Shared Pool-u onda baza podataka neće preduzimati dodatne korake za optimizaciju i izvršenje.

# Ulazni tokeni

Ulazni tokensi predstavljaju osnovne gradivne elemente SQL upita. Tokeni su fragmenti teksta koji predstavljaju pojedinačne delove upita, kao što su ključne reči, identifikatori, operatori i konstante. Na primer, u SQL upitu prikazanom na slici 3 ulazni tokeni su “SELECT”, “FirstName”, “LastName”, “FROM”, “Employee”, “WHERE”, “Age”, “<”, “30”, “AND”, “Department”, “=”, “Developers”.

A screenshot of a computer

Description automatically generated

Slika 3. Primer upita

# Generisanje sintaksnog stabla (syntax tree)

Tokom faze parsiranja, čoveku čitljivi oblik upita se transformiše u oblike koje je moguće koristiti unutar sistema za upravljanje bazama podataka (DBMS). Ovi oblici mogu biti izrazi relacione algebre, stabla upita ili grafovi upita. Ova transformacija omogućava DBMS-u da efikasno manipuliše upitima i izvršava ih, prilagođavajući se specifičnostima podataka i zahtevima korisnika.

Na primeru, za SQL upit prikazan na slici 4:



Slika 4. Primer SQL upita za koji se generiše stablo

Sintaksno stablo će izgledati ovako:



Slika 5. Sintaksno stablo

Rezultat faze pasiranja je sintaksno stablo.

# Sintaksna provera

Sintaksna provera je važan korak u analizi upita. Ovde se vrši provera ispravnosti sintakse SQL upita kako bi se identifikovale eventualne greške i nepravilnosti. SQL Server proverava da li su sve ključne reči, identifikatori, operatori ispravno napisene i da li upit sledi pravila SQL sintakse. Ako se pronađe bilo kakva sintaksna greška, SQL Server će generisati odgovarajuću grešku ili upozorenje.

Na slici 6 je prikazan primer gde upit nije uspešno izvršen jer je korišćena pogrešna sintaksa.

A screenshot of a computer

Description automatically generated

Slika 6. Pogrešna sintaksa prilikom izvršenja upita

Kao što se i očekivalo, izvršenje upita je zaustavljeno jer SQL Server nije uspeo da prepozna ključnu reč “WHERE” zato što je pogrešno napisana.

# Semantička provera

Semantička provera je utvrđuje da li je naredba smislena, odnosno SQL Server identifikuje koje tabele i kolone su navedene u upitu i proverava njihovu validnost.

Na primer, ako napišemo SQL upit sa netačno napisanim imenom kolone, slika 7:

A screenshot of a computer

Description automatically generated

Slika 7. SQL naredna sa netačnim imenom kolone

SQL Server će identifikovati grešku zbog netačno napisanog imena kolone "LastName", odnosno utvrdiće da kolona “Last” ne postoji u tabeli “Employee”.

Analiza se vrši upoređivanjem celog teksta upita sa podržanim ključnim rečima za verziju baze podataka koja se trenutno koristi.

Na slici 8 je prikazana uspešno izvršena SQL naredba.

A screenshot of a computer

Description automatically generated

Slika 8. Uspešno izvršena SQL naredba

# Shared Pool provera

Shared Pool check je korak u procesu obrade upita u MSSQL bazi podataka koji se odnosi na proveru da li već postoji optimizovani plan izvršenja za dati SQL upit u deljenom memorijskom segmentu poznatom kao “shared pool”.

Baza podataka koristi algoritam heširanja za generisanje heš koda za svaku SQL naredbu. Heš kod naredbe je SQL ID (jedinstveni identifikator). Kada korisnik unese SQL naredbu, baza podataka će prvo generisati hash kod za taj upit. Hash kod se generiše na osnovu samog teksta upita, tako da dva identična upita daju isti hash kod. Nakon generisanja hash koda, baza podataka će proveriti da li već postoji optimizovani Execution plan za taj hash kod u Shared pool-u. Ovaj deo memorije čuva optimizovane Execution planove za nedavno izvršene SQL upite radi ponovne upotrebe. Ako se pronađe optimizovani plan za dati hash kod u shared pool-u, baza podataka će koristiti taj plan umesto ponovnog generisanja novog plana. To znači da se SQL upit neće ponovo parsirati, optimizovati i izvršavati, već će se koristiti već postojeći Execution plan.

Na primer, ako izvršimo sledeći upit:

SELECT \* FROM Employee WHERE Department = 'Developers'

Baza podataka će generisati hash kod za ovaj upit i proveriti da li već postoji optimizovani plan izvršenja za taj hash kod u Shared pool-u. Ako postoji, baza podataka će koristiti taj plan. Na primer, ako je već ranije izvršen sličan upit koji se odnosi na Department 'Developers', baza podataka može koristiti već postojeći plan izvršenja umesto ponovnog generisanja novog plana. Ovo ubrzava obradu upit.

U zavisnosti od toga da li će se koristiti već postojeći Execution plan ili generisati novi, postoje dve kategorije parsiranja - Hard parse i Soft parse:

* Hard parse – predstavlja slučaj kada ne postoji heš kod za upit i ne može da se iskoristi već postojeći Execution plan, onda upit mora da prođe kroz dodatne korake: Optimizer i Row Source Generation.
* Soft parse – predstavlja slučaj kada postoji heš kod za upit koji se izvršava, pa nije potrebno prolaziti kroz dodatne korake već se može iskoristi postojeći Execution plan.

Soft parse je poželjnije od Hard parse jer baza podataka preskače korake optimizacije i row source generation-a, i direktno prelazi na izvršenje.

# Optimizacija upita

Optimizacija upita je proces odabira efikasnog Execution plana za izvršenje upita. Nakon parsiranja upita, parsirani upit se transformiše u formu koja je obično stablo upita ili graf i prosleđuje se optimizatoru upita, koji generiše različite Execution planove za izvršenje parsiranog upita i odabir plana sa najmanjim procenjenim troškovima. Catalog manager pomaže optimizatoru da izabere najbolji plan za izvršavanje upita generišući troškove svakog plana.

# Generisanje različitih planova izvršavanja (Execution planova)

Proces odabira jednog plana izvršenja od potencijalno mnogo mogućih planova se naziva optimizacija upita. Optimizator upita je jedna od najvažnijih komponenti Database Engine-a. Dok optimizator upita koristi neke dodatne troškove za analizu upita i odabir plana, ovi troškovi se obično višestruko čuvaju kada optimizator upita izabere efikasan plan izvršenja. Na primer, dve građevinske kompanije mogu dobiti identične nacrte za kuću. Ako jedna kompanija provede nekoliko dana na početku da planira kako će izgraditi kuću, a druga kompanija počne da gradi bez planiranja, kompanija koja odvoji vreme da planira svoj projekat verovatno će prva završiti.

SQL Server Optimizator upita je optimizator zasnovan na troškovima. Svaki mogući plan izvršenja ima pridružene troškove u smislu količine korišćenih računarskih resursa. Optimizator upita mora analizirati moguće planove i izabrati onaj sa najnižim procenjenim troškovima. Neke složene SELECT naredbe imaju hiljade mogućih planova izvršenja. U ovim slučajevima, optimizator upita ne analizira sve moguće kombinacije. Umesto toga, koristi složene algoritme da pronađe plan izvršenja čija je cena razumno blizu minimalnoj mogućoj ceni.

SQL Server Optimizator upita ne bira samo plan izvršenja sa najnižim troškovima resursa, već bira plan koji vraća rezultate korisniku uz razumnu cenu resursa i koji najbrže vraća rezultate. Na primer, paralelna obrada upita obično koristi više resursa nego serijska obrada, ali se upit završava brže. Optimizator upita će koristiti paralelni plan izvršavanja da bi vratio rezultate ako to neće negativno uticati na opterećenje servera.

Optimizator upita se oslanja na statistiku distribucije kada procenjuje troškove resursa različitih metoda za izdvajanje informacija iz tabele ili indeksa. Statistika distribucije se čuva za kolone i indekse i sadrži informacije o gustini osnovnih podataka. Ovo se koristi za označavanje selektivnosti vrednosti u određenom indeksu ili koloni. Na primer, u tabeli koja predstavlja automobile, mnogi automobili imaju istog proizvođača, ali svaki automobil ima jedinstveni identifikacioni broj vozila (VIN). Indeks na VIN-u je selektivniji od indeksa na proizvođaču, jer VIN ima manju gustinu od proizvođača. Ako statistika indeksa nije aktuelna, optimizator upita možda neće napraviti najbolji izbor za trenutno stanje tabele.

Gustina definiše distribuciju jedinstvenih vrednosti koje postoje u podacima, ili prosečan broj dupliranih vrednosti za datu kolonu. Kako se gustina smanjuje, selektivnost vrednosti se povećava.

Optimizator upita je važan jer omogućava serveru baze podataka da se dinamički prilagođava promenljivim uslovima u bazi podataka bez potrebe za unosom od programera ili administratora baze podataka. Ovo omogućava programerima da se fokusiraju na opisivanje konačnog rezultata upita.

Recimo da imamo sledeći SQL upit koji treba da izabere imena i prezimena zaposlenih iz tabele “Employee” koji imaju više od 23 godina i pripadaju “Developers” department-u:



Slika 9. Primer SQL upita

SQL Server će generisati različite planove izvršavanja koji mogu uključivati različite strategije pristupa podacima i indeksa. Na primer, moguće je da SQL Server odabere plan izvršavanja koji koristi indeks na koloni “Department” i “Age” kako bi ubrzao izvršavanje upita. Nakon toga, SQL Server će odabrati najefikasniji plan izvršavanja na osnovu analize cene izvršavanja i drugih faktora kako bi obezbedio brze i efikasne rezultate.

# Row Source Generation

Row source generator je softver koji prima optimalni plan izvršenja od optimizatora i proizvodi iterativni plan izvršenja koji može da koristi ostatak baze podataka.

Iterativni plan je binarni program koji, kada se izvrši od strane SQL engine-a, proizvodi skup rezultata. Plan ima oblik kombinacije koraka. Svaki korak vraća skup redova. Sledeći korak ili koristi redove u ovom skupu, ili poslednji korak vraća redove aplikaciji koja izdaje SQL naredbu.

Row source je skup redova koji vraća korak u planu izvršenja zajedno sa kontrolnom strukturom koja može iterativno obraditi redove. Izvor reda može biti tabela ili rezultat operacije pridruživanja ili grupisanja.

# Execution plan

Plan izvršenja SQL upita (Execution plan) je mapa koju baze podataka koristi za efikasno izvršavanje upita. To je strategija korak po korak koja opisuje kako će baza podataka obraditi upit, uključujući koji će se indeksi ili table scans koristiti, redosled operacija i kako će se podaci spajati i filtrirati.

SQL Server Management Studio ima tri opcije za prikaz planova izvršenja:

* The Estimated Execution Plan (Procenjeni plan izvršenja) - ovaj tip plana izvršenja je samo nagađanje Optimizatora upita kako će se konkretni koraci izvršiti prilikom vraćanja rezultata. Često se generiše pre nego što se upit izvrši.
* The Actual Execution Plan (Stvarni plan izvršenja) – ovaj plan se generiše nakon što je upit izvršen. Prikazuje stvarne operacije i korake uključene tokom izvršavanja upita. On se može ili ne mora razlikovati od procenjenog plana izvršenja.
* The Live Query Statistics (Statistika upita uživo) - je isti kao Actual plan plus njegov kontekst izvršenja. Uključuje informacije o vremenu izvođenja tokom napretka izvršavanja i one se ažuriraju svake sekunde. Informacije o vremenu izvršavanja uključuju, na primer, stvarni broj redova koji prolaze kroz operatore.

Postoje tri različita formata pregleda planova izvršenja u SQL Server Management Studio:

1. Grafički
2. XML
3. Tekstualni

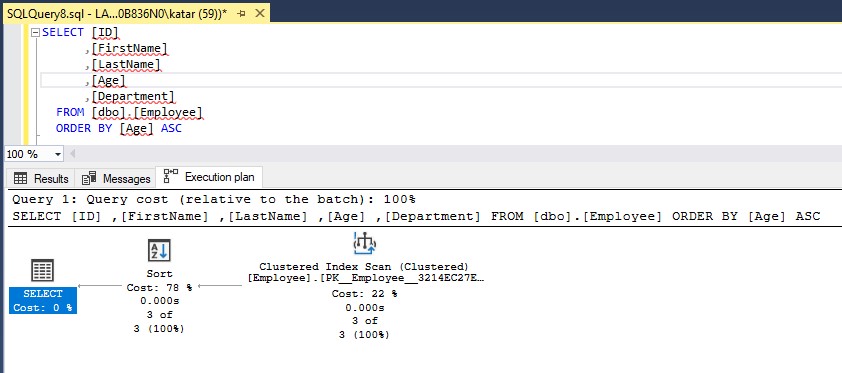
SSMS podrazumevano generiše plan izvršenja u grafičkom formatu.

Kada se generišu planovi izvršenja, prikazaće se dijagrami, slike 10 i 11.

A screenshot of a computer

Description automatically generated

Slika 10. Estimated Execution plan



Slika 11. Actual Execution Plan

U planu izvršenja prikazanom na gornjoj slici 11, ako se pređe kursorom preko komponenti, mogu se videti detaljne statistike za svaku od operacija i komponenti koje se prikazuju u planu izvršenja. Plan se tumači zdesna nalevo i odozgo prema dole. Pošto se ovaj konkretan plan sastoji od samo jednog reda, nema potrebe za pristupom od vrha do dna.

Interno, SQL Server izvršava plan s leva na desno. Svaki operater traži podatke od sledećeg operatera. Na primer, u gornjem scenariju, Top operator zahteva podatke od Clustered Index Scan operatora. Clustered Index Scan operator vraća redove iz tabele Employee.

Plan izvršenja sa slike se može podeliti na pet dela:

* Clustered Index Scan
* Data Flow from Clustered Index Scan (strelica)
* Sort Operator
* Data Flow from Sort Operator (strelica)
* Select Operator

Ako plan izvršenja čitamo s desna na levo, primetićemo sledeće:

Prvi operator je Clustered Index Scan koji čita podatke iz tabele Employee.

Kada se zadrži pokazivač miša na Clustered Index Scan operatoru, biće prikazani detaljni rezultati, slika 12.

A screenshot of a data analysis

Description automatically generated

Slika 12. Prikaz detaljnih rezultata operatora Clustered Index Scan

Na gornjoj slici se mogu videte sledeći detalji:

1. Physical Operation: Physical Operations su objekti koji obavljaju fizičke operacije. Neki od primera su Index Seek, Clustered Index Scan itd. Logički operatori daju uputstva ovoj vrsti operatora da izvrši definisane operacije.
2. Logical Operation: Koristi se kod Physical Operation-a. Takođe daje jasnu sliku o tome koji upit je potrebno obraditi i kako će se on obraditi.
3. Actual Execution Mode: Prikazuje stvarni plan koji treba da se izvrši. Koristi ga procesor za obradu i za izvršavanje upita.
4. Estimated Execution Mode: Sličan je Actual Execution Mode-u, jedina razlika je što prikazuje procenjenu vrednost.
5. Storage: Kazuje kako će optimizator upita skladištiti rezultate koji se dobiju izvršenjem upita.
6. Estimated Operator Cost: Ovo nije stvarna cena, već relativna informacija u odnosu na druge operatere u planu izvršenja.
7. Estimated I/O Cost: Pokazuje cenu ulazno/izlaznih operacija skupa rezultata.
8. Estimated CPU Cost: Procenjuje cenu za izvršenje operacija sa CPU-om.
9. Estimated Subtree Cost: Cena stabla izvršenja koja se trenutno čita zdesna nalevo i odozgo na dole.
10. Number of Executions: Prikazuje broj izvršenja koje optimizator može da obradi u jednoj seriji.
11. Estimated Number of Execution: Slično Number of Executions-u, ali jedina razlika je u tome što će dati procenjenu vrednost.
12. Estimated Number of Rows per Execution: Procena optimizatora koliko će redova biti vraćeno.
13. Estimated Number of Rows to be Read: Procena optimizatora koliko će redova biti pročitano.
14. Estimated Row Size: Veličina skladištenja svakog reda u operatoru.
15. Ordered: Određuje da li je u skupu podataka nad kojem se operacija izvodi implementirano sortiranje ili ne.
16. Node ID: Automatsko dodeljivanje broja po redosledu kojim se operater poziva u planu izvršenja čitajući zdesna nalevo i odozgo nadole.

Pored toga, postoje još dve komponente:

1. Object: Definiše tabelu nad kojom je izvršena operacija.
2. Output List: Definiše izabrane kolone koje će biti prikazane u skupu rezultata.

Kada se postavi pokazivač miša na strelicu koja povezuje Clustered Indek Scan i Top operator, prikazuje se procenjeni i stvarni broj redova, i procenjena veličinu podataka.

A screenshot of a computer

Description automatically generated

Slika 13. Procenjeni i stvarni broj redova, i procenjena veličinu podataka

Na slici 14 je prikazan Live Execution Plan.

A screenshot of a computer

Description automatically generated

Slika 14. Live Execution Plan

Live Execution Plan daje procenu upita uživo zajedno sa protokom podataka i procentima dovršenosti pojedinačnih operatera. Obično se čeka da se završi izvršenje upita kako bi se video stvarni plan izvršenja. Ali, pretpostavimo da je upitu potrebno više od 30 minuta da dobije rezultate. U ovom slučaju, može da se koristi statistika upita uživo da bi se pratili pojedinačni operatori, tok podataka i kome operateru treba vremena da dovrši upit.

Na slici 15 se može videti Execution plan za upit sa slike u XML format, a na slici 16 u tekstualnom formatu.

A screenshot of a computer

Description automatically generated

Slika 15. Execution plan u XML formatu

A screenshot of a computer

Description automatically generated

Slika 16. Execution plan u tekstualnom formatu

Execution planovi SQL upita pružaju detaljan uvid u složeni svet optimizacije performansi baza podataka. Razumevanje strukture, izvršavanja i optimizacije SQL upita je ključno za programere i administratore koji imaju za cilj da kreiraju sisteme baza podataka visokih performansi. Razotkrivanjem tajni Execution planova, može se postići izuzetno korisničko iskustvo i pojednostavljene operacije nad podacima.

# Zaključak

Proces obrade upita kod MSSQL baze podataka predstavlja ključni deo u radu sa podacima i omogućava efikasno izvršavanje SQL upita radi dobijanja željenih rezultata. Kroz niz koraka, uključujući parsiranje, analizu, optimizaciju, izvršavanje i vraćanje rezultata, baza podataka obrađuje SQL upite korisnika. Ovaj proces je od značaja za performanse i pouzdanost sistema jer utiče na brzinu izvršavanja upita, optimizaciju resursa i upravljanje pristupom podacima. Ovaj rad istražio je ključne faze procesa obrade upita, počevši od parsiranja i analize, preko optimizacije i izvršavanja, sve do vraćanja rezultata korisniku.

Tokom parsiranja, baza podataka proverava sintaksu i semantiku upita i pretvara ga u strukturu podataka koja može dalje da se procesuira. Nakon toga, optimizacija upita ima za cilj generisanje najefikasnijeg plana izvršenja, uzimajući u obzir faktore kao što su indeksi, veličina podataka i dostupni resursi. Na kraju sledi izvršavanje upita i vraćanje rezultata.

U današnjem digitalnom svetu, brz pristup i efikasno upravljanje podacima su od suštinskog značaja za uspešno poslovanje i donošenje odluka. Kroz kontinuirano usavršavanje i primenu najboljih praksi u procesu obrade upita, organizacije mogu osigurati konkurentsku prednost i unaprediti svoje poslovanje.

# Literatura

Intro to SQL Server Architecture -

<https://www.geeksforgeeks.org/intro-to-sql-server-architecture/?ref=ml_lbp>

SQL Server -

<https://www.datacamp.com/tutorial/sql-server-tutorial>

MS SQL Server Tutorials Point -

<https://www.tutorialspoint.com/ms_sql_server/ms_sql_server_tutorial.pdf>

What is SQL and what is it used for? -

<https://www.futurelearn.com/info/blog/what-sql-used-for>

What is SQL (Structured Query Language)? -

<https://pwskills.com/blog/sql/>

SQL Server Query Processing -

<https://logicalread.com/sql-server-query-processing-w01/>

Query Processing -

<https://www.geeksforgeeks.org/sql-query-processing/>

SQL Processing -

<https://docs.oracle.com/en/database/oracle/oracle-database/19/tgsql/sql-processing.html#GUID-8CF633B1-EAC4-47C7-9189-C479ADEF1FFA>

Query Optimization -

<https://www.du.ac.in/du/uploads/departments/Operational%20Research/24042020_querry%20processing%20and%20optimization%20and%20centralized%20Vs%20distributed%20database.pdf>

Query processing architecture -

<https://learn.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide?view=sql-server-ver16>

Execution Plans in SQL Server -

<https://www.sqlshack.com/execution-plans-in-sql-server/>

What is an Execution plan in SQL Server and how to use it? -<https://www.tatvasoft.com/blog/optimize-sql-query/>

Understanding SQL Query Execution Plans - <https://divyanshbhatia.medium.com/understanding-sql-query-execution-plans-unveiling-the-path-to-database-performance-ad9aee04b9f7>