Tallinna Ülikool

Informaatika Instituut

MSSQL 2012 Administreerimine. Õppematerjal.

Bakalaureusetöö

Autor: Sander Kuusk

Juhendaja: Jaagup Kippar

Tallinn 2014

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Sisukord

Sissejuhatus

Paljud andmebaasidega tegelevad inimesed kasutavad *Microsoft SQL Server*'i (edaspidi *MSSQL*) tarkvara. Väga sageli ei osata ära kasutada võimalusi, mis aitavad andmebaasi sisse ehitatud programmidega efektiivselt ära kasutada, nii talitab ka autor ja tema tuttavad. Samuti ei ole teoreetiline osa andmebaaside administreerimisest piisavalt selge. Näiteks ei osata indekseid päringutes korrektselt kasutada ning automatiseerida lihtsamaid toimingud andmebaasides. Selle asemel on võimalik aega ja vaeva kokku hoida, kasutades *MSSQL* sisse ehitatud administreerimise vahendeid.

Töö autor valis bakalaureusetöö jaoks antud teema, kuna omab kogemust *Microsoft*'i poolt toodetud andmebaasi vahenditega, kuid ei tunne vahendeid põhjalikult, mida *Microsoft* oma andmebaasi serverile lisanud on. Töö autor leiab, et need vahendid väärivad tutvustamist.

Käesoleva bakalaureusetöö eesmärgiks on anda ülevaade mõningatest MSSQL administreerimist abistavatest vahenditest ning koostada õppematerjal. Töö autor märgib, et osa harjutustest on laetud üles www.sandersql.eu keskkonda.

Eesmärkide saavutamiseks autor:

- Uurib MSSQL administreerimise vahendeid;
- Paneb paika õppematerjali sisu;
- Loob ja analüüsib õppematerjali;
- Viib läbi õppematerjali testimise;

Seega on antud Bakalaureusetööst kasu inimestel, kes huvituvad MSSQL andmebaasi administreerimisest, kuid omavad väheseid kogemusi selles valdkonnas.

Käesolev bakalaureusetöö jaguneb kaheks peatükiks. Esimeses peatükis annab töö autor ülevaate Microsoft SQL Serveri andmebaasidest, kus leiab vastused küsimustele. Mis on MSSQL? Mis versioonid on olemas? Teine peatükk annab ülevaate õppematerjali ettevalmistustest, sisaldab õppematerjali teoreetilist osa. Samuti saavad vastused küsimustele. Mida peab üks andmebaasi administraator (*DBA – Database Administrator*) teadma tööks? Millised on head tavad, millest tasub kinni pidada?

Töö lõppu on lisatud õppematerjal (Lisa 1 – Õppematerjal)

1. MSSQL tutvustus ja ajalugu

Peatüki esimeses osas on selgitav ülevaade Microsoft SQL andmebaasi serverist ja teine osa kirjeldab lühidalt *MSSQL* tehnoloogia ajalugu.

1.1 MSSQL tutvustus

MSSQL (*Microsoft Structured Query Language*) on *Microsoft* poolt toodetud relatsiooniline andmebaasi haldamise süsteem (*RDBMS – Relational Database Management System*). Tegemist on täieliku tootega, mis on peamiselt disainitud konkureerima *Oracle* andmebaasi süsteemiga ja *MySQL*'iga. *MSSQL* on nagu enamus *SQL* Servereid, mis toetavad *ANSI SQL*'i. Viimaseks nimetatakse standard *SQL* keelt. Samuti toetab rakendus *Transact SQL*'i (*T-SQL),* mis on Microsoft enda nägemus *SQL* skriptimiskeelest.

Lisaks on Microsoft loonud mitmeid tööriistu SQL Serveri haldamiseks. Aastate jooksul on need saanud mitmeid vajalikke parandusi ja täiendusi.

MSSQL Serveri tööriistad:

- *MSSQL*'i põhi tööriistaks on *SQL Server Management Studio* (SSMS), millega koostatakse skripte. Töö autor mainib ära, et SSMS toetab nii 32-bit kui ka 64-bit'iseid keskkondi.
- *SQL Server Profiler* võimaldab administraatoril jälitada (*Trace*) erinevaid SQL Serveri tegevusi, et hiljem oleks vigade otsimine kergem (*Troubleshoot*).
- *SQL Server Database Engine Tuning Advisor* (DETA) annab administraatorile soovitusi, et parandada SQL Serveri üleüldist jõudlust (*Performance*).
- *SQL Server Configuration Manager* (SSCM) kasutatakse SQL Serveri teenuste konfigureerimiseks. Näiteks kuidas kliendi programm teeb ühenduse SQL Serveri pihta.
- Käsurea (*Command Prompt*) tööriistad osql.exe ja sqlcmd.exe. Viimane laseb kasutajal käivitada T-SQL teksti, süsteemi protseduure ja skripti faile otse käsurealt.
- *SQL Server Data Tools* (SSDT) lisa *plugin*'ad *Microsoft Visual Studio*'le.

SQL Server ei koosne vaid andmebaasi mootorist vaid seal on mitmeid erinevaid komponente, mis võimaldavad kasutajatel luua, hallata ja analüüsida rakendusi.

MSSQL Serveri komponendid (Microsoft, 2014):

- Andmebaasi mootor (*Database Engine*)
- Analüüsi teenused (*Analysis Services*)
- Integratsiooni teenused (*Integration Services*)
- Raportite teenused (*Reporting Services*)
- Ärianalüüsi teenused (*Master Data Services*)
- CEP platvorm (*StreamInsight*)
- Ärianalüüs (*Data Mining*)
- Täistekstiotsing (*Full-Text Search*)
- (*Power Pivot*)
- Andmebaaside replikeerimine (*Replication*)
- Andmete kvaliteedi teenused (*Data Quality Services*)
- (*Power View*)

Microsoft on välja lasknud SQL Server 2012 mitu erinevat versiooni. Kõik need versioonid omavad ühtemoodi relatsioonilist andmebaasimootorit, kuid erinevad üksteisest kas riistvara ressursside või lisavõimaluste poolest. (Microsoft, 2014)

- *Enterprise Edition* – sisaldab kogu SQL Serveri funktsionaalsust ja on mõeldud suurtele ettevõtetele, mis kasutavad keerulisi ärilahendusi.
- *Standard* – sisaldab põhilisi SQL Serveri funktsioone ja on mõeldud väikestele ja keskmistele ettevõtetele.
- *Business Intelligence* – sisaldab ärianalüüsi vahendeid.
- *Express* – SQL Serveri tasuta versioon õppimiseks, arenduseks ja väikeste nõudmistega ettevõtetele.
- *Web* – mõeldud veebihaldajatele
- *Developer* – mõeldud ainult arendustööde tegemiseks.

1.2 MSSQL ajalugu

SQL server on olnud kättesaadav juba mitmeid aastaid, kuid mille arenemine ei ole jäänud seisma. MSSQL'i ajalugu sai alguse juba aastal 1989 kui Microsoft väljastas versioon 1.0 SQL serverist. Versioon 1.0 ja 1.1 vajasid OS/2 operatsiooni süsteemi, kuid 4.2 ja edasi liikusid juba *Windows*'i platvormile. Versioon 7.0 loodi OLAP (*Online Analytical*

Processing) teenused, mis hiljem muutusid Analüüsi teenusteks (*Analysis Services*). SQL Server 2000 tõi juba mitme instantsi (*instance*) ja järjestuse (*collation*) võimaluse, samuti ärianalüüsi Office programmide jaoks (*data mining*) ja 64-bit tugi. SQL Server 2005 vahetas välja senise administratsiooni tööriistad *SQL Server Management Studio* vastu. Tutvustati dünaamilisi haldus vaateid (*Dynamic Management Views*). SQL server 2008 tõi *FileStream*'i kättesaadavaks ning spetsialiseeritud kuupäeva ja aja tüübid. SQL Server 2008 R2 parandas oluliselt vigu mis esinesid SQL Server 2008's. SQL Server 2012 ühendas omavahel äritarkvara (*Business Intelligence)* projektid *Visual Studio 2010*'ga.

Järgnevalt toob töö autor välja kõik MSSQL versioonid ja väljalaske aasta (Tabel 1 Microsoft SQL Serveri versioonid ja väljalaskeaastad)

Versioon	Aasta	Versioon	Aasta
1.0	1989	2000	2000
1.1	1991	2005	2005
4.2	1992	2008	2008
4.21	1994	2008R2	2010
6.0	1995	2012	2012
6.5	1996	2014	2014
7.0	1998		

Tabel 1 Microsoft SQL Serveri versioonid ja väljalaskeaastad

2. Õppematerjali ülevaade

Peatüki esimeses osas antakse ülevaade käesoleva bakalaureusetöö raames koostatud õppematerjali ettevalmistustest. Teine peatüki sisaldab õppematerjali analüüsi. Kolmandast osast leiab lugeja õppematerjali teoreetilise osa. Samuti annab töö autor näpunäiteid kuidas materjali paremini omastada. Iga peatükki lõppu on lisatud hea praktika, mida võiks algaja administraator teada.

2.1 Ettevalmistused

Enne õpematerjali koostamist oli vaja leida vastused mitmele küsimusele:

- Miks on vaja luua MSSQL 2012 administreerimise õppematerjal?
  - Õppematerjali loomise vajadus seisnes MSSQL Serveri tehnoloogiat puudutava eestikeelse informatsiooni puuduses. Kuna töö autor kasutab igapäevaselt antud tarkvara ning otsib inglisekeelseid juhendeid, siis tekkis idee koostada ise vastav eestikeelne õppematerjal.
- Kes on sihtgrupp ja mis on nende oskused antud valdkonnas?
  - Töö autor arvab, et kõige enam huvituvad SQL Serveri administreerimisest inimesed, kes on eelnevalt kokku puutunud SQL süntaksiga ning on soov õppida administreerimist. Seega eeldab õppematerjali kasutamine mõningasi algteadmisi SQL süntaksist.
- Mida sisaldab Õppematerjal?
  - Õppematerjal sisaldab: MSSQL Server 2012 tutvustust. Samuti seletab töö autor erinevaid andmebaasi taastamise mudeleid (*Recovery Models*), kuidas automatiseerida SQL Serveri tegevusi. Peale eelneva leiab bakalaureusetööst andmebaasi halduse peatüki, mis seletab lahti indeksite olemuse ja tähtsuse andmebaasis. Andmebaasi administraator peaks samuti teadma kuidas jälgida tegevusi SQL Serveris ning kuidas SQL tegevus logisid hiljem analüüsida. Lisaks illustreerib töö autor päringute analüüsimist ning nende muutmist efektiivsemaks.
- Kes võiksid Õppematerjali testida?
  - Õppematerjali testijate otsimisel tuli arvestada asjaoluga, et töö autori poolt loodud õppematerjal eeldab mõningasi algteadmisi andebaasidest.

- Töö autor valis õppematerjali testijateks tuttavad.

2.2 Õppematerjali analüüs

Õppematerjal valmis *Microsoft SQL Server 2012* tarkvara põhjal, sest autori soov oli anda ülevaade ühest populaarseimast relatsioonilise andmebaasi süsteemist, mis on kõigile huvilistele kättesaadav (*SQL Server 2012 Express*).

Koostatud õppematerjal on mõeldud peamiselt *MSSQL Server*'i huvilistele ning eeldab algteadmisi andmebaasidest. Kuna eeldatud algteadmiste kohta leidub nii Internetis, raamatukogudes kui ka mujal piisavalt eestikeelset informatsiooni iseõppimiseks, siis autor leidis, et nende teemade selgitamine õppematerjalis ei ole vajalik. Samuti on olemas Jaagup Kippari poolt loodud SQL materjal algajatele (Kippar, 2011) Sellest tulenevalt peaks lugeja oskama alla laadida *SQL Server Express* tarkvara ning lisama sinna Microsofti poolt toodetud näite andmebaasi (*AdventureWorks2012*). Samuti on programmi *SQL Server Management Studio* (SSMS) põhikäskluste teadmine vajalik.

Õppematerjali loomisel lähtus autor enda kogemustest MSSQL Server 2012 programmiga, üritades samal ajal arvestada, et mitmed antud bakalaureusetöö lugejad ei ole piisavalt kokku puutunud relatsiooniliste andmebaasidega. Eesmärk oli koostada võimalikult lihte ja arusaadav õppematerjal, mida oleks nii mugav jälgida kui järgida.

Õppematerjali läbis kaks inimest, kes andsid tagasisidet suuliselt.. Mõlemad kasutajad läbisid õppematerjali töö autori arvutil, kus olid kõik ettevalmistused tehtud. Kokkupuude andmebaasidega oli mõlemal inimesel võrdlemisi väike, kuid töö autor viis nad kurssi andmebaasi teooriaga. Mõlemal kasutajal läks üle tunni aja aega õppematerjali läbimiseks, peamiselt kulus aega ülesannetest arusaamiseks. Õppematerjali positiivse poole pealt toodi välja lihtsus ja põhjalikkus, kuid miinusena, et oleks võinud olla veel ülesandeid.

2.3 Õppematerjali kirjeldus

Töö autor pidi koostama õppematerjali, mille põhjal on võimalik tutvuda MSSQL 2012 halduse jaoks vajalike teemadega. Õppematerjali sisu ja ülesanded peavad olema hea ja demonstreerima lugejatele erinevate andmebaasi haldamise punktide vajalikkust. Õppematerjal hõlmab juhiseid Andmebaasi serveri haldamiseks ning on jaotatud viieks suuremaks osaks.

Esimene peatükk „Taastamise mudelid (*Recovery models*)“ kirjeldab erinevaid taastamise strateegiaid samuti kui kaua võib taastamine aega võtta ning kui palju andmeid võib kaduma minna.

Teine peatükk „SQL Serveri automatiseerimine (*Automating SQL Server Management*)“ kirjeldab kuidas automatiseerida tegevusi kasutades SQL Server Agenti samuti kuidas hallata SQL Server agenti.

Kolmas peatükk „Andmebaasi hooldus (*Database maintenance*)“ kirjeldab kuidas tagada andmete terviklikkuse andmebaasis ning kuidas hallata indekseid.

Neljas peatükk „SQL Serveri jälgimine (*Monitoring SQL Server*)“ kirjeldab võimalust kuidas jälgida hetkeseisu serveris, püüda kinni lühiajaliselt andmeid, hallata neid ning hiljem analüüsida.

Viies peatükk „SQL Serveri järg (*SQL Server trace*)“ kirjeldab kuidas püüda kinni andmeid pikaajaliselt ning neid hiljem analüüsida.

Kuues peatükk „SQL Serveri päringute optimeerimine (*Optimize SQL Queries*)“ kirjeldab kuidas optimeerida ja analüüsida päringuid.

2.3.1 Taastamise mudelid (*Recovery models*)

SQL Serveril on kolm andmebaasi taastamise mudelit. Kõik mudelid hoolitsevad selle eest, et säilitada andmed katastroofi korral. Siiski on kõik kolm mudelit väga erinevad ning andmebaasi administraator peab teadma, milline taastamise mudel sobib tema andmebaasile kõige paremini. Andmebaasi taastamise mudel valikul peab administraator järgmisi faktoreid arvestama: (Microsoft, 2012)

- Varukoopiate intervall
- Võimalik andmekadu
- Millised varukoopia võimalused on mulle saadaval

Küsides ükskõik millise äriomaniku käest, kui palju andmeid võivad nad kaotada, siis reeglina vastatakse, et ei tohi kaotada ühtegi andmehulka, ükskõik mis asjaoludel. See on eesmärk, mida iga andmebaasi administraator üritab, kuid kahjuks pole see reaalne eesmärk, mida saavutada. Selletõttu on kaks aspekti, mis vajavad läbimõtlemist. Esiteks, kui kaua võib varukoopiast taastamine aega võtta. Teiseks kui suur võib olla andmekadu. (Microsoft, 2012)

- RTO (*Recovery Time Objective*) – Ei ole otstarbekas omada ideaalset varukoopiat, kui sellest andmete kättesaamine võtab liiga kaua aega. Seega peab varukoopia strateegia omama RTO'd. Töö autor märgib, et kõige targem on omada plaani, mis sisaldab kiiret varukoopiat koos väikese andmete kaoga. Samuti võib suuremates organisatsioonides kohata probleemi, kus on vaja leida esmalt kasutaja, kellele on võimaldatud vastavad ligipääsud varukoopiatele ja dokumentatsioonile jne.
- RPO (*Recovery Point Objective*) – Kui süsteem on taastatud edukalt, loodetavasti mõistliku aja jooksul, tekib järgmine küsimus. Kui palju andmeid on kaduma läinud? Töö autor leiab, et päevane kadu on mõistlik, kuid mõne ettevõtte jaoks ei ole ka see aktsepteeritav ning nii tehakse andmebaasidest lausa iga tunni tagant koopiaid.

Samuti peab andmebaasi administraator arvesse võtma andmebaasi suurust. Järgnevalt tutvustab töö autor kolme andmebaasi taastamise mudelit: (Microsoft, 2012)

- Lihtne taastamise mudel (*Simple Recovery Model*) vähendab administraatori tööd kuna viimane ei lase kannete (*Transactional*) logist varukoopiat teha. Lihtne taastamise mudel on seetõttu väga riskantne kuna andmete kadu võib olla suur andmebaasi rikke korral. Andmed on taastatavad vaid viimase varukoopia tegemiseni ning seetõttu peaksid andmebaasid, mis kasutavad lihtsat taastamise mudelit, olema seadistatud lühikese intervalli tagant tegema varukoopiat. Samas peab olema intervall piisavalt pikk, et see ei segaks töö tegemist.
- Täielik taastamise mudel (*Full Recovery Model*) pakub mõistlikku andmebaaside hoolduse mudelit, kus väljaannete (*transactions*) vastupidavus on vajalik. SQL Serveri installimisel on määratud vaikimisi kõikidele andmebaasidele täielik taastamise mudel. Täieliku taastamise mudelil on vajalikud logi varukoopiad. See taastamise mudel logib kõik väljaanded ja hoiab neid väljaande logis kuniks nendest on tehtud varukoopia. Täielik taastamise mudel lubab taastada andmebaasi täpselt ennem katastroofi, eeldusel, et logi lõpuosast on võimalik teha koopia peale riket. Samuti lubab see mudel taastada terve andmebaasi spetsiifilise ajahetkeni (*point-in-time*) ning individuaalseid andmete lehti (*data pages*)
- *Bulk-logged Recovery Model* võimaldab vähendada kannete logi nõudeid mitmele massi (*bulk*) operatsioonile. See mudel on mõeldud peamiselt täieliku taastamise mudeli abistamiseks. Näiteks, kui andmebaasis teostada suuremat massi importi või indeksite loomist siis tasub kaaluda andmebaasi ajutist muutmist *Bulk-logged*

taastamise mudeli peale. See ajutine pöörang lubab andmebaasi sooritusvõimel tõusta kuna logitakse vaid määratud jaotisi (*allocations*) ja logi ruumi kasutus väheneb. Samuti hoiab *bulk-logged* taastamise mudel väljaannete logi senikaua kuni nendest on tehtud varukoopia. Selle mudeli miinused on suurenenud logide varukoopiate maht ja suurenenud andmete kadumise võimalused, sest *bulk-logged* taastamise mudel ei toeta andmebaasi taastamist spetsiifilise ajahetkeni.

Taastamise mudel	**Kirjeldus**
Lihtne	• Ei luba logist varukoopiate tegemist • Automaatselt kärbib logi, et ruumi nõuded oleksid võimalikult väikesed
Täielik	• Vajab logide varukoopiaid • Hoiab ära andmekao, mis võiks olla põhjustatud katkisest või kaotatud andmefailist • Võimaldab taastada andmebaasi teatud hetkeni ajas.
Bulk-Logged	• Vajab logide varukoopiaid • Võimaldab tõsta jõudlust tänu mass (*bulk*) kopeerimistegevustele • Vähendab logi ruumi kasutust kuna kasutab minimaalset logimist mass kopeerimisoperatsioonidele

Tabel 2 Taastamise mudelid (Microsoft, 2012)

Järgnevalt tutvustab töö autor kõiki varukoopiate tüüpe, sest kõik varukoopia tüübid ei ole saadaval kõikidele andmebaasi taastamise mudelitele. Näiteks kannete logi varukoopiat ei ole võimalik teha andmebaasist, millel on lihtne taastamise mudel. (Microsoft, 2012)

Varukoopiate tüübid

- *Full* – Kõik andmefailid ja aktiivne osa transaktsiooni logist.
- *Differential* – Varukoopia tehakse vaid osadest, mida on muudetud pealt viimast täis varukoopia tegemist.
- *Partial* – Varukoopia tehakse peamisest faili grupist, igast loe/kirjuta faili grupist ja vajadusel mõnest täpsustatud faili grupist.

- *Transaction Log* – Varukoopia tehakse igast muudatusest mis on salvestatud logidesse.
- *Tail-Log Backup* – Logi varukoopia tehakse vaid logi „sabast“ just enne taastamise (*restore*) funktsiooni.
- *File/File Group* – Varukoopia tehakse vaid täpsustatud failidest või failigruppidest.
- *Copy Only* – Varukoopia tehakse kas andmebaasist või logist.

Järgnevalt toob töö autor välja erinevad plaanid, kuidas teostada varukoopia planeerimist

- Täieliku andmebaasi varukoopia planeerimine. on sobilik väiksemahulistele andmebaasidele, mida on võimalik kiiresti varundada. Siiski, kui andmebaasi mahud kasvavad, siis samuti kasvavad ka varukoopiate maht ja aeg.
- Kannete põhise logi varukoopia planeerimine on sobilik andmebaasidele, kus tehakse pidevaid modifikatsioone. Kannete logi varukoopia tegemisel salvestatakse kõik andmed kuni eelmise logi varukoopiani.
- Diferentsiaalse varukoopia planeerimine on sobilik suuremahulistele andmebaasidele kuna see võimaldab kiirendada varukoopiate tegemist ning vähendada andmekadu.

Varukoopia kompressiooni tutvustati esmakordselt *SQL Server 2008 Enterprise* versioonis. Kuna kokku pakitud varukoopia on palju väiksem, kui tavaline varukoopia, aga andmete maht on sama. Samuti kasutab kokku pressitud versioon vähem seadme I/O ja seetõttu on see reeglina kiirem. Samas antud tegevus kurnab oluliselt rohkem protsessorit (CPU) ning see tegevus võib avaldada mõju SQL Serveri jõudlusele. Seega on targem teha kokk pakitud varukoopiat ajal, mil serveris ei ole suurt koormust. (Microsoft, 2012)

Hea praktika:

- Planeeri oma varukoopiate strateegiat ettevaatlikult.
- Mõista äri vajadusi varukoopiate strateegia planeerimisel.
- Vali sobiv varukoopia mudel.
- Planeeri kannete logi varukoopia tiheduse põhjal kannete logi varukoopia.
- Tasub mõelda diferentsiaal varukoopia variandi peale, kuna see suurendab kogu süsteemi võimsust.

2.3.2 SQL Serveri automatiseerimine (*Automating SQL Server Management*)

Aegajalt puutub andmebaasi administraator kokku protsessidega, mis juhtuvad korduma. Andmebaasi administraator võib lõigata palju kasu erinevate protsesside automatiseerimisega. Enamus neist haldus protsessidest keskenduvad usaldusväärsele ja järjekindlale rutiinile. SQL Server pakub mitmeid võimalusi oma tegevuste automatiseerimiseks, kui põhiline tööriist on SQL Server Agent. Kõik andmebaasi administraatorid peaksid olema tuttavad *SQL Server Agent*'i konfiguratsiooni ja kestva haldusega. (Microsoft, 2012)

SQL Server Agent töötab kahel tähtsal põhimõttel. Esiteks tööd mis on mõeldud automatiseerimiseks (Microsoft, 2012)

- Vähendatud administratsiooni koorem (*Reduced Administrative Workload*). Näiteks võib mõne ülikooli Administraator saada mitmeid taotlusi uute kasutajakontode loomiseks. Viimane võib täiesti vabalt ükshaaval need luua kasutades standardseid töövahendeid, kuid efektiivsem administraator õpib skripti kirjutama ja loob kasutaja kontod läbi selle, sest mõni koodirida on lihtsam käivitada.
- Usaldusväärne ülesannete teostamine (*Reliable Execution of Routine Tasks*). Kui rutiinsed tegevused käivitada käsitsi siis on alati risk, et midagi jääb kahe silma vahele Näiteks kui Andmebaasi administraator unustab pühapäeva õhtul andmebaasist varukoopia tegemise. Automatiseerimine lubab administraatoritel fokusseerida vaid eranditele, mis ilmnevad rutiinse tegevuse käigus.
- Järjekindel ülesannete teostamine (*Consistent Execution of Routine Tasks*). Veel üks probleem, mis võib juhtuda rutiinsete tegevuste käsitsi tegemisega on see, et need ei pruugi iga kord sama moodi käituda. Näiteks kui andmebaasi administraatorilt palutakse arhiveerida osa andmeid produktsiooni tabelitest arhiivi tabelitesse igal Esmaspäeva hommikul. Uutel tabelitel peab olema sama nimi nagu originaal tabelitel, kuid lisandiga, et lõpus on sufiks mis koosneb jooksvast päevast. Administraator võib isegi mäletada, et ta peab iga esmaspäeva hommikul seda protsessi kordama, kuid tõenäosus, et ta eksib mõnega järgmistest on võrdlemisi suur.
    - Kopeerib valed tabelid.
    - Kopeerib vaid mõned tabelid.
    - Unustab lisada õige päeva sufiksisse.

- Vormindab valesti kuupäeva, mis asub sufiksis.
- Kopeerib andmed valesse arhiivi tabelisse.

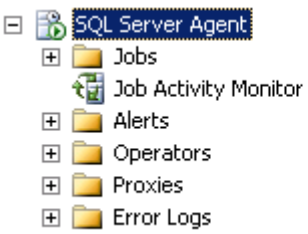
Kõik, kes on kokku puutunud käimasolevate administratsiooni süsteemidega siis võivad öelda, et sellised probleemid tulevad aeg ajalt ette. Automatiseerimine tagab, et tegevused toimuksid järjekindlalt just siis kui nad käivitatakse.

- Ennetav haldussüsteem (*Proactive Management*). Kui rutiinsed tegevused on juba automatiseeritud siis võib kergesti tekkida situatsioon, kus rutiine tegevus annab mingil põhjusel veateate, kuid administraator ei pane seda tähele. Näiteks on andmebaasi administraatorid automatiseerinud igapäevase andmete varundamise, kuid ei ole märganud, et tegevus ei ole jõudnud korrektse lõpuni. Saabub hetk, kus läheb vaja varukoopiast taastamist, kuid ühtegi töötavat varukoopiat enam võtta pole. Seetõttu tuleb luua teadete teatamise (*notifications*) süsteem, mis annab administraatorile teada, kui mõni tegevus ei ole korrapäraselt lõpuni jõudnud.

Ülevaade *SQL Server Agent*'ist

- Töötab, kui Windowsi teenus
- Peab jooksma, et
  - Käivitada töid
  - Vallandada häireid
  - Luua ühendust operaatoritega.
- Agenti käivitus peaks olema automaatne

Töö autor märgib, et SQL Server 2012 Express versioonis ei ole võimalik käivitada SQL Server Agenti, sest selline funktsionaalsus on tasulistel versioonidel. Joonis 1 SQL Server Agent SQL Server 2008R2 versioonis



Joonis 1 SQL Server Agent SQL Server 2008R2 versioonis

Hea praktika:

- Planeeri rutiinsed tööd kasutades SQL Agenti.
- Loo isetehtud kategooriad, et grupeerida oma töid
- Loo oma töödest skript, et neid oleks võimalik kasutada ka mujal süsteemides.
- Kasuta töö ajalugu, et analüüsida, kas töö oli edukas või mitte.
- Kasuta töö aktiivsuse monitori (*Job Activity Monitor*), et näha reaalajas tööde tegemist.

2.3.3 Andmebaasi hooldus (*Database maintenance*)

Andmebaasi rikutus on võrdlemisi haruldane nähtus, kuid andmebaasi administraatori jaoks on see üks tähtsamaid ülesandeid, mida kontrollida. Riknenud failide taastamine sõltub, kui kiiresti need üles leitakse. SQL Serveri indeksid töötavad ilma igasuguse hoolduseta, kuid aegajalt vajavad need puhastamist, sest võivad olla killustatud (*fragmentation*). SQL Server omab hooldus plaani rakendust (*Maintenance Plan Wizard*), et abistada hooldustööde rutiinset loomist. (Microsoft, 2012)

Sellest peatükist leiab vastused järgnevatele küsimustele:

- Kuidas tagada andmebaasi terviklikkus?
- Kuidas hallata indekseid?
- Mida tähendab hoolduste automatiseerimine?

Võrdlemisi haruldane on nähtus, kus andmebaasi mootor põhjustab otseselt andmete rikutust (*corruption*). Samas tuleb meeles pidada, et andmebaasi mootor sõltub suuresti riistvaralisest platvormist, mis seda kõike hoiab töös ning see võib põhjustada andmete riknemist. Konkreetsemalt öeldes probleemid mäludega ning sisendi ja väljundiga võivad viia andmebaasi riknemisele. Kui andmebaasi administraator ei avasta riknemist varakult võib see viia oluliselt suuremate probleemide juurde. Selleks, et märgata riknemist andmebaasis loodi käsk *DBCC CHECKDB*. (Microsoft, 2012)

Ülevaade käsust DBCC CHECKDB

- Kontrollib loogilist ja füüsilist andmebaasi terviklikkust
    - Kõikide lehtede (*pages*) paigutust andmebaasis;
    - Tabelite ja indeksite konsistents;
    - Kataloogi konsistents andmebaasis;
    - Lingi taseme konsistents *Filestream* objektidel;

    - Teenuste vahendaja (*Service Broker*) objektidel;
- Pakub parandus võimalusi;
    - Mõned võimalused lubavad andmete kadu;
- Kasutab sisemist andmebaasi hetketõmmist (*Snapshot*);
- Käsklust peaks tihti tööle panema, et saada varakult jaole andmete riknemisele;

DBCC tähendab andmebaasi terviklikkuse kontrollimine (*Database Consistency Checker*). CHECKDB valikuvõimalus DBCC utiliidis teeb võimalikuks eriti sügava uuringu andmebaasi struktuuris, et otsida võimaliku riknemist.

Valik	Kirjeldus
DBCC CHECKALLOC	Kontrollib määratletud andmebaasi struktuuri jaotust ketta ruumi suhtes
DBCC CHECKTABLE	Kontrollib spetsiifilise tabeliga seotud lehti (*pages*) ja viitab lehtede vahele, mis on seotud tabeliga. DBCC CHECKDB teostab DBCC CHECKTABLE käsu iga tabeli kohta andmebaasis
DBCC CHECKCATALOG	Kontrollib andmeid andmete kohta andmebaasis. Need tabelid hoiavad informatsiooni süsteemi tabelite, kasutaja tabelite ja andmebaasi objektide kohta. Ei kontrolli kasutaja loodud tabeleid (*user table*)

Tabel 3 DBCC CHECKDB valikud (Microsoft, 2012)

Praktika on näidanud, et on hea *DBCC CHECKDB* lasta käima just ennem andmebaasi varukoopia tegemist. See aitab kindlustada, et varukoopia sisaldab terviklikke andmeid. Samuti sisaldab antud käsklus mitmeid erinevaid valikud, mis muudavad käskluse käitumist. (Microsoft, 2012)

- *PHISICAL_ONLY* valikut kasutakse tihti produktsiooni süsteemide peal, sest see vähendab oluliselt *DBCC CHECKDB* ajakulu suurte andmebaaside peal. Töö autor märgib, et siiski peaks regulaarselt tööle laskma ka täisversiooni *CHECKDB*'st
- NOINDEX valik võimaldab täpsustada, et intensiivseid kontrolle ei tehta *nonclustered* indeksitele, mis asuvad kasutajate loodud tabelites. See vähendab samuti üleüldist

käskluse teostamise aega, kuid ei mõjuta süsteemseid tabeleid kuna rikkumatuse (*integrity*) kontrollid teostatakse alati süsteemsete tabelite indeksite peal. NOINDEX valik annab võimaluse uuesti üles ehitada *nonclustered* indeksid, juhul kui nad peaksid muutuma rikutuks.

- EXTENDED_LOGICAL_CHECKS valik teostab detailse kontrolli CLR kasutaja loodud andmete tüüpidele ja ruumilistele (*spatial*) andmetüüpidele.
- TABLOCK valik paneb iga tabeli lukku seniks ajaks kuni CHECKDB teeb konsistentsi kontrolli. Kasutab andmebaasi hetketõmmiste asemel lukke.
- ALL_ERRORMSGS tagastab kasutajale kõik veateated, muidu tagastab vaid esimesed 200.
- NO_INFOMSGS tagastab kasutajale kõik veateatud ja mitte ühtegi informatsiooni teadet.
- ESTIMATEONLY valik hindab *tempdb* ketta ruumi suurust.

Lisaks kõikidele eelnevatele veateadetele pakub DBCC CHECKDB ka kahte parandus valikut. Töö autor märgib, et andmebaasis peab sisse lülitatud olema *single user mode*. (Microsoft, 2012)

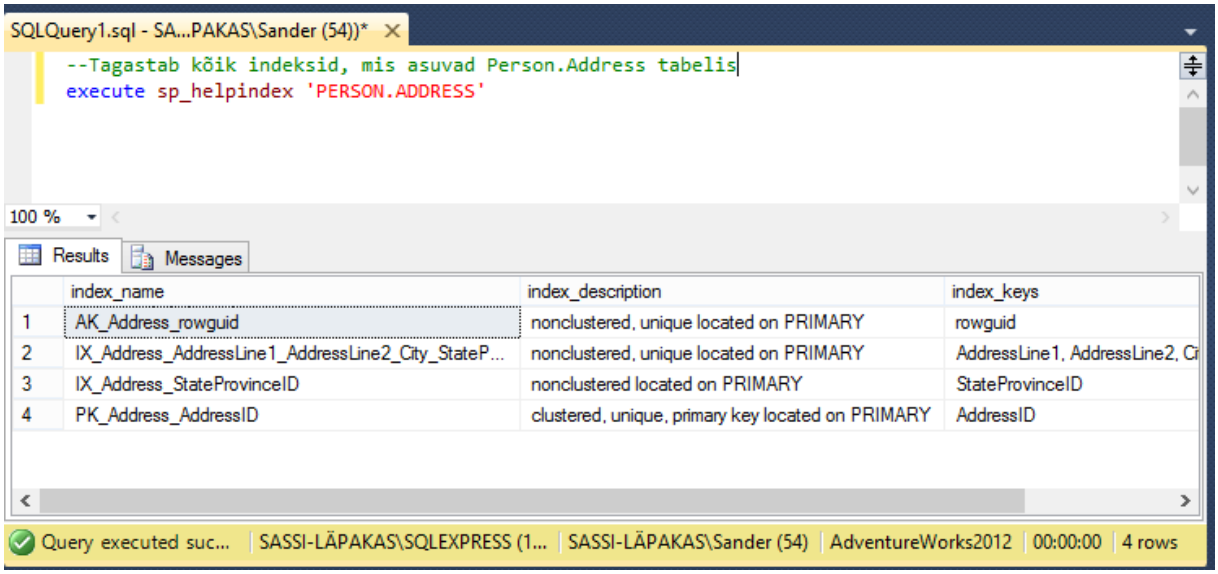
- REPAIR_REBUILD valik ehitab uuesti indeksid ja katkised andmete lehed eemaldatakse. See töötab vaid kergetel riknenud failidega ja ei sisalda andmete kadu.
- REPAIR_ALLOW_DATA_LOSS valik sisaldab alati andmete kadu. See eraldab riknenud lehed ja muudab teisi lehti mis viitavad riknenud lehtedele. Kui operatsioon on edukalt lõpule viidud on andmebaas terviklik, kuid seda vaid füüsilisi andmebaasi rikkumatuse vaatevinklist.

Ükskõik, mis põhjusel peaks SQL Server vajama ligipääsu andmetele, mis asuvad tabelis siis andmebaasi mootor teeb otsuse kas lugeda kõiki lehti (*Pages*) või äkki on mõni indeks tabelis, mis vähendaks päritava rea leidmise aega. Indeksite kasutamine ei ole kohustuslik, kuid kui päring peab lugema suurel hulgal lehti, siis teeb see tulemuse kättesaamise aeglasemaks. Indeksid muudavad otsimise, sorteerimise ja liitumise (*Join*) protsessid oluliselt kiiremaks. Samas vajavad nad korrapärast hooldust ja lisa kettaruumi. Vahetevahel SQL Server loob ise omale ajutisi indekseid, et parandada päringu kiirust. Ajutisi indekseid kasutatakse ainult siis kui õiget indekseerimist ei eksisteeri. (Microsoft, 2012)

Clustered Indeks – Tabel, kus esineb rühmitatud (*Clustered*) indeks, omab eelhäälestatud ridade järjestust lehel ja lehed omakorda tabelis. Järjestus põhineb võtmel, mis luuaks ühest või mitmest veerust. Seda võtit nimetatakse rühmade võtmeks. Ainult üksainus rühmitatud indeks saab tabelis eksisteerida kuna tabelis esinevad read on ainult ühes järjestuses. Rühmitatud indeksitel on alati indeks id = 1. Rühmitatud indeks on kiirem kui rühmitamata indeks, sest viimane peab tagasi tabelile viitama, kui selekteeritud veergu ei ole indekseeritud. (Microsoft, 2012)

NonClustered Indeks – Rühmitamata indekseid võib olla tabelil piiramatult. Indeks ja andmed on salvestatud erinevatesse kohtadesse. Indeks sisaldab andmete asukoha viitasid. Rühmitamata indeksiga võib tuua paralleele raamatuga, kus raamatu alguses on sisukord, mis viitab teatud andmetele. Indeks ise on organiseeritud kas kasvavas või kahanevas järjestuses.

Hea viis kontrollida indeksite olemasolu ja tüüpi on kasutada järgmist savestatud protseduuri (*Stored Procedure*) „*Execute sp_helpindex tabel*“, mis tagastab kõik indeksid antud tabelis. Joonis 2 *sp_helpindex* kasutamine. (Microsoft, 2012)



Joonis 2 *sp_helpindex* kasutamine

Indeksite fragmenteerimine on probleem, millega tuleb andmebaasi administraatoril aeg ajalt tegeleda. Fragmenteeritus esineb siis, kui SQL Server tunneb ära indeksite lehe, kus andmete muutus põhjustas indeksite lehe poolitamise. Esineb kahte sorti fragmenteeritust sisemine ja välimine. (Microsoft, 2012)

- Sisemine (*Internal*) esineb siis, kui lehel ei hoiata nii palju andmeid, kui nad võimaldaksid ehk lehed on pooltühjad. Seda juhtub kui sisestamise (*Insert*) operatsiooni käigus leht poolitatakse või uuendamise (*Update*) käigus põhjustatakse rea liikumist teisele lehele.
- Välimine (*External*) killustatus esineb, kui on lehti, mis on loogiliselt järjestatud, kuid lehe numbrid on järjestamata. Kui läheb vaja uut indeksi lehte sisestada, siis see paigutatakse loogiliselt õigesse kohta, kuid võidakse paigutada ka indeksite listi lõppu.

FILLFACTOR ja *PAD_INDEX* pakuvad andmebaasi administraatorile võimaluse anda vaba ruumi indeksite lehele. *FILLFACTOR* = 80 tähendab seda, et leht läheb 80% täis ja 20% on vaba igast lehest.

SQL Serveri administraatoril on kaks võimalust kuidas eemaldada fragmentatsioon rühmitatud ja rühmitamata indeksitest. (Microsoft, 2012)

- Indeksite uuesti üles ehitamine (*Rebuild*) kustutab ja loob uued indeksid. See eemaldab fragmentatsiooni ja nõuab tagasi (*Reclaim*) ketta ruumi. Kui lisavalik ALL on valitud, siis kustutakse ja luuakse uued indeksid terve tabeli ulatuses ühe operatsiooniga. Indeksite uuesti üles ehitamise operatsioon võtab palju kannete logi ruumi kuna tegevus logitakse sinna. Töö autor märgib, et vaba kettaruumi peab olema piisavalt, et indeksid uuesti üles ehitada. Sobilik väga fragmenteerunud indeksite (> 30%) jaoks
- Indeksite ümber korraldamine (*Reorganize*) kasutab minimaalselt süsteemi ressursse. Organiseerib ümber lehtede loogilise järjestuse. Lisaks ümber korraldamine surub kokku indeksite lehed. Kokku surumine võtab arvesse *FILLFACTOR*'i väärtuse.

Üks põhilisi ülesandeid, mida andmebaasi administraator peab tegema SQL Serveris on statistika uuendamine. Seda on vaja teha, et päringud oleksid optimeeritud. SQL Server peab otsustama, mis indekseid ta peab vastava päringu juures kasutama. Need otsused tehakse statistika põhjal. Statistika uuendab üldjoontes SQL Server ise, kuid *AUTO_UPDATE_STATISTICS* valik peab olema võimaldatud (*Enabled*). (Microsoft, 2012)

Statistikat on võimalik uuendada ka käsu peale. Selleks on vaja käivitada UPDATE STATISTICS tabeli vastu ja kõik statistika, mis on selle tabeliga seotud uuendatakse.

Samuti on võimalik kasutada süsteemi sisest salvestatud protseduuri *sp_updatestats*, mis uuendab kõik statistika andmebaasis. (Microsoft, 2012)

Hea praktika:

- Jooksuta regulaarselt käsklust DBCC CHECKDB.
- Sünkroniseeri DBCC CHECKDB oma varukoopia strateegiaga.
- Tasub mõelda RESTORE käskluse peale ennem, kui andmebaasis esineb rikutust.
- Defragmenteeri indeks seal, kus vaja.
- Uuenda statistikat regulaarselt.
- Kasuta Hoolduse Plaani (*Maintenance Plan*), et lisada sinna regulaarseid ülesandeid.

2.3.4 SQL Serveri jälgimine (*Monitoring SQL Server*)

Microsoft SQL Serveri andmebaasi mootor on võimeline hakkama saama ilma administratsioonita pikka aega. Samas jälgides igapäevaselt serveri seisu on võimalik ennetada tulevasi veateateid juhul, kui need peaksid ilmuma. SQL Server pakub mitmeid tööriistu serveri hetkeseisu jälgimiseks. Andmebaasi administraator peab teadma neid ja oskama nendega ümber käia. (Microsoft, 2012)

Alates versioonist SQL Server 2005 on kasutusel dünaamilised halduse vaated (*Dynamic Management Views*) ja dünaamilised halduse funktsioonid (*Dynamic Management Functions*).

DMV'd ja DMF'd tagastavad kasutajale serveri oleku informatsiooni. Antud informatsiooni analüüsides saab diagnoosida probleeme, parandada serveri sooritus võimet jne. On olemas kahte sorti DMV'si ja DMF'si.

- Serveripoolsed
- Andmebaasipoolsed

Kõik DMV'd ja DMF'd eksisteerivad *sys* skeemis ja järgivad nime *dm_%*. DMV'd ja DMF'd on kategoriseeritud järgmiselt.

Kategooria	Kirjeldus
Sys.dm_exec_%	Need objektid pakuvad informatsiooni ühenduste, sessioonide, taotluste (*request*) ja päringute kohta. Näiteks, *sys.dm_exe_sessions* annab iga avatud

	sessiooni kohta rea päringu vastuses.
Sys.dm_os_%	Need objektid pakuvad informatsiooni SQL operatsioonisüsteemi kohta. Näiteks, *sys.dm_os_performance_counters* pakub ligipääsu SQL serveri loenduritele kasutamata operatsiooni süsteemi tööriistu.
Sys.dm_tran_%	Need objektid pakuvad informatsiooni kannete (*transaction*) haldamise kohta. Näiteks, *sys.dm_os_tran_active_transactions* pakub detailset ülevaadet hetke kannetest.
Sys.dm_io_%	Need objektid pakuvad informatsiooni I/O protsesside kohta. Näiteks, *sys.dm_io_virtual_file_stats* pakub detailset informatsiooni ja statistikat I/O sooritusvõime kohta.
Sys.dm_db_%	Need objektid pakuvad informatsiooni andmebaasi kohta. Näiteks, *sys.dm_db_index_usage_stats* pakub informatsiooni selle kohta kuidas iga indeks andmebaasis on kasutatud.

Tabel 4 DMV kategrooriad (Microsoft, 2012)

DMV'de list on nähtaval avades *Object Explorer* ja sealt liikudes edasi *System Views*

*DMF'*ide list on nähtaval avades master andmebaasi, sealt liikudes edasi *System Functions.*

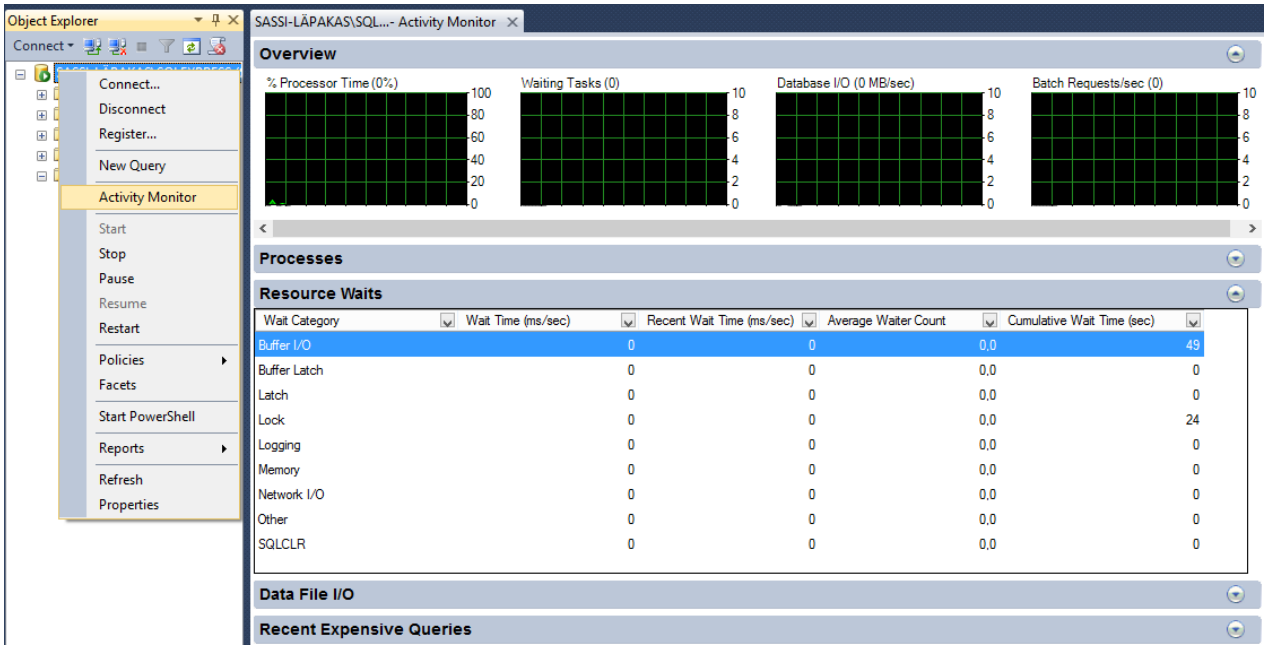
On olemas kahte tüüpi dünaamilisi haldamise objekte:

- Objektid, mis tagastavad informatsiooni reaalaja kohta süsteemis.
- Objektid, mis tagastavad informatsiooni viimase ajaloo põhjal.

Näide 1 reaalajas informatsiooni tagastamise kohta. Kaks DMV'd ühendatakse. *Sys.dm_exec_sessions* tagastab iga sessiooni kohta ühe rea ja *sys.dm_os_waiting_tasks* tagastab iga ootelisti (*waiting on resource*) kohta ühe rea. Kombineerides need vaated (*views*) ja lisades filtri saab leida kasutajate ülesannete ootelisti asukoha, mis on oodanud rohkem kui näiteks 3000 millisekundit.

Näide 2 viimase ajaloo põhjal informatsiooni tagastamise kohta. Sys.dm_os_wait_stats tagastab kasutajale informatsiooni, kui tihiti ja kui kaua on ülesanne pidanud ootama spetsiifilise *wait_type* järgi kuni SQL Serveri instants on tööd alustanud.

Aktiivsuse monitor (*Activity monitor*), mis asub *SQL Server Management Studios* pakub mitmeid lisavõimalusi jälgimaks andmebaasi serveri hetke seisu. Viimane näitab informatsiooni SQL Serveri protsesside, ootamiste (*Waits*), I/O ja kulukate päringute kohta. Samuti nõuab aktiivsuse monitor VIEW SERER STATE luba, et vaadata sealt andmeid. Lisaks on võimalik sealt ka eemaldada protsesse, mis blokeerivad teisi. Joonis 3 *Activity monitor* kasutamine (Microsoft, 2012)



Joonis 3 *Activity monitor* kasutamine

Hea praktika:

- Kasuta dünaamilisi hallatavaid objekte, et sooritada reaalajas serveri jälgimist ja veateadete otsimist.
- Kasuta aktiivsuse monitori, et saada ligi reaalajas andmetele.
- Kasuta *Performance Monitor*'i, et koguda andmeid Windowsi ja SQL Serveri kohta.
- Loo *Management Data Warehouse*, mis hoiaks informatsiooni jõudluse ajaloo kohta.

2.3.5 SQL Serveri jälg (*SQL Server Trace*)

Suurte organisatsioonide jaoks ei ole Microsoft SQL Server sooritusvõime parandamine ühekordne tegevus. Reeglina on see pikk protsess, mis pidevalt parandab SQL serveri sooritusvõimet. Võimalus jälitada tegevusi annab tarkvaraarendajale võimaluse hiljem

tulemusi analüüsida, et suunata oma jõud juba spetsiifiliste osade parandamiseks. (Microsoft, 2012)

Protsesside jälitamiseks kasutatakse *SQL Server Profiler*'it ja *Extended Events Profiler*'it. Andmebaasi mootori häälestamise nõustaja (*Database Engine Tuning Advisor*) aitab analüüsida eelnevalt kinni püüdud andmeid. Samuti pakub rakendus parandusi, mida administraator võiks teha, et päringud jookseksid kiiremini. Reeglina on parandused seotud indeksite lisamise ja statistikaga. (Microsoft, 2012)

SQL Server Profiler püüab kinni andmed sündmuste (*events*) kohta kui need peaksid tekkima. Administraator saab valida sündmusi, mida hakatakse kinni püüdma. Jälg (*trace*) sisaldab vaid eelvalitud sündmusi. Samuti on olemas eelseadistatud mallid (*template*) ning võimalik on ka enda loodud mallide salvestamine hilisemaks kasutamiseks. Lisaks on saab ka laadida SQL *Server Profiler*'isse *Windows Performance Monitor* logisid. See lubab veelgi põhjalikumat analüüsi luua. (Microsoft, 2012)

Kui *SQL Server Profiler* jälg on aktiivne, siis kogutakse kõik kinni püüdud andmed kokku ja kuvatakse läbi graafilise kasutajaliidese kasutajale. Lisaks on võimalik püütud sündmused salvestada kas operatsiooni süsteemi failina või andmebaasi tabelisse.

Salvestatud informatsioon on võimalik jagada kategooriateks. Viimased sisaldavad sündmusi, mis on defineeritud läbi erinevate tulpade. Mingi tegevuse ilmumine SQL Serveri andmebaasi mootoris tähistab sündmust. Sündmused on omakorda defineeritud läbi tunnuste (*attribute*)

Sündmus	Kirjeldus
SQL :BatchCompleted	Kui partii (*batch*) T-SQL lõpetab siis antud sündmus esineb logis.
SQL StmtCompleted	Kui partiist pole võimalik aru saada, siis on võimalik välja otsida iga partii individuaalse lause detailid.
RPC: Completed	Ilmneb, kui salvestatud protseduur lõpetab tegevuse.
Audit Login. Audit Logout	Ilmneb igal sisse ja välja logimisel.
Deadlock Graph	Püüab kinni kõik sundlukud (*Deadlock*) ja nende detailid.

Tabel 5 Populaarseimad sündmused mida jälgitakse (Microsoft, 2012)

SQL Trace	*SQL Server Profiler*

• Kasutab salvestatud protseduure töötamiseks;
• Töötab andmebaasi mootori sees;
• Kirjutab sündmused failidesse või tabelitesse;
• Kasutatakse;
  o Pikaajaliseks jälgimiseks;
  o Andmebaasi sooritusvõime parandamiseks;
  o Suuremahuliste jälgede salvestamiseks;
	• Kasutatakse graafilise töövahendina;
	• Võtab kasutusse *SQL Trace;*
	• Kirjutab kas failidesse või andmebaasi tabelitesse;
	• Kasutatakse;
	  o Vigade kõrvaldamiseks test süsteemides;
	  o Lühiajaliseks analüüsimiseks;
	  o Väiksema mahuliste jälgede salvestamiseks;

Tabel 6 *SQL Trace* ja *SQL Server Profiler* võrdlus (Microsoft, 2012)

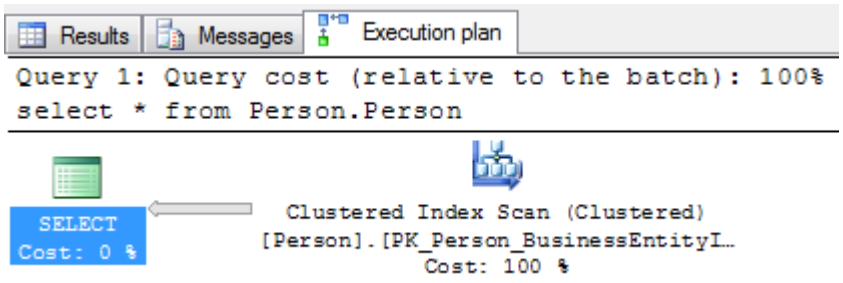
Hea praktika

- Kasuta *SQL Server Profiler*'it ajaliselt lühikeste jälgede jaoks
- Kasuta *SQL Trace*'it suurte ja ja kauakestvate jälgede jaoks.
- Kasuta *SQL Server Profiler*'it, et defineerida jälgi ja koostada skript nendest *SQL Trace* jaoks.
- Impordi jälgede andmed andmebaasi tabelitesse põhjalikumaks analüüsiks.
- Kasuta *Database Engine Tuning Advisor*'it, et analüüsida andmebaasis esinevaid töövooge.

2.3.6 SQL Serveri päringute optimeerimine (*Optimize SQL Queries*)

Andmebaasi administraatoril tuleb aegajalt ette probleeme individuaalsete päringute kiirusega, mis omakorda võib muude programmide tööd häirida. Selleks tuleb vaadata üle päringud, mis lähevad andmebaasi pihta. Isegi kui andmebaasi server jookseb kõige võimsamal riistvaral siis võib selle jõudlus olla negatiivselt mõjutatud tänu mõnele halvale koostatud päringule. Üleüldist andmebaasi jõudlust on võimalik tõsta, kui parandada ka kõige ressursinõudlikumaid päringuid või päringuid, mida käivitatakse kõige tihedamini. (Microsoft, 2012)

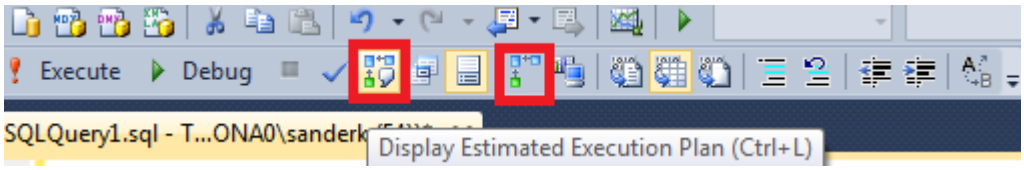
Kui on plaanis hakata parandama üksikut päringut, siis tuleks alustada päringu teostamise plaaniga (*Execution Plan*). Viimane kirjeldab protsesse, nii füüsilisi kui ka loogilisi, mida SQL Server käivitab, et viia täide päring. Teostamise plaani koostab andmebaasi mootori komponent nimega *Query Optimizer*. Lisaks võetakse arvesse mitmeid erinevaid faktoreid: otsingu predikaatidega, mida kasutatakse tabelites, tabelite ühendamised (*Join*), mis veerud tagastatakse ja kas on võimalik indekseid ära kasutada. (Microsoft, 2012)



Joonis 4 *Actual Execution Plan*

Keerulisemate päringute jaoks ei otsi komponent välja kõiki erinevaid võimalusi, vaid otsib plaani, mis sobib piisavavalt hästi antud päringu jaoks. Permutatsioone võib esineda keerulises päringus väga palju ning seetõttu ei otsigi *Query Optimizer* välja kõiki plaane. (Microsoft, 2012)

- *Management Studio* pakub *Display Actual Execution Plan* ja *Display Estimated Execution Plan* võimalusi. Mõlemate plaanide tulemused on graafilised. Samuti on need ka kõige populaarsemad plaanid, mida analüüsitakse.



Joonis 5 *Display* ja *Actual Estimated Execution Plan*

- *SET* valikud nagu näiteks *SHOWPLAN_XML* ja *SHOWPLAN_ALL* tagastavad kasutajale teostamise plaani kas *XML* kujul või läbi tekstirea.
- *SQL Server Profiler* sündmuste klassid nagu näiteks *Showplan XML* võimaldab kasutajal koguda kõik teostamise plaanid tänu jälje (*Trace*) aktiviseerimisele.

XML tüüpi fail ei pruugi olla kõige lihtsamini loetav, kuid see annab võimaluse kirjutada protseduure ja vahendeid analüüsimaks teostuse plaane. Samuti saab XML põhjal faili salvestada *.sqlplan* laiendiga, et hiljem avada need hiljem uuesti *Management Studio*

programmis. See on väga kasulik, kui on soov hiljem võrrelda kuidas ajaga on plaanid muutunud.

Esimese asjana tuleks aru saada, kuidas teostamise plaanid koostatakse. SQL Server kasutab kulupõhist (*cost-based*) päringu optimeerimist, see tähendab, et genereeritakse teostamise plaan, mis on kõige madalama kuluga. Eelkalkulatsioon põhineb iga tabeli statistikal, mis on kättesaadav optimeerijale. Kui selline statistika on puudu või iganenud, siis päringu optimeerijale ei jõua kohale elutähtis informatsioon, mida tal läheks vaja. Suure tõenäosusega selle tõttu on hinnang päringule vale. Sellistel puhkudel valib optimeerija vähem optimaalsema plaani see tähendab, et teostuse kulu on kas ülehinnatud või alahinnatud erinevate plaanide lõikes.

Paratamatult tehakse valesid oletusi eelkalkulatsiooni teostuse kulule. Esiteks, on see piisavalt dokumenteeritud, et mis üksused (*Units*) väljendavad eeldatavat maksumust ja kas neil on otsene seos (*Relation*) täitmise ajal. Teiseks, kuna see on hinnanguline ja see võib olla vale, siis planeeritust suuremad kulud võivad osutuda oluliselt tõhusamaks CPU, I/O ja täitmise (*Execution*) ajal. Seda võib juhtuda tabeli muutujatega, sest nende jaoks ei ole statistikat ning päringu optimeerija arvab, et see sisaldab vaid ühte rida. Tegelikult sisaldab see kordades ridu.

Üks spetsiaalne teostuse plaan on paralleelne plaan. Juhul, kui Serveril on rohkem kui üks protsessor siis võidakse valida paralleelne plaan päringu käivitades. Paralleelne plaan „maksab“ rohkem, et käivituda ja seda on võimalik näha eelkalkulatsiooni teostuse ajal. Tänu suurenenud töötluse võimsusele on võimalik paralleelsed plaanid kiiremini lõpuni viia kui standard plaanid. Administraator saab ise valida maksimaalse parallelismi valiku serveri tasemel ja selle vajadusel üle kirjutada kasutades päringus OPTION(MAXDOP n).

Kokkuvõte

Käesoleva bakalaureusetöö põhieesmärkideks oli anda ülevaade Microsoft SQL Server 2012 administreerimiseks vajalikest rakendustest ning koostada sellekohane õppematerjal. Lisaks anda algajale andmebaasi administreerijale näpunäiteid.

Töö autor annab kirjanduse põhjal ülevaate osa Microsoft SQL Server 2012 komponentidest, mida peab algaja administraator haldama. Veel on töös välja toodud erinevad parimad praktikad mida võiks administraator teada.

Üheks eesmärgiks oli anda algajale andmebaasi administraatorile teavet, mida vaadata, otsida, analüüsida hallates *Microsoft SQL Server 2012*. Töö autor koostas õppematerjali, kus on õpetlikke harjutusi, mida administraator võiks läbida. Lisaks andis töö autor omapoolseid juhiseid, kuidas efektiivsemalt kasutada teatud olukordades relatsioonilist andmebaasimootorit.

Õppematerjali testiti kahe kasutaja poolt, kes andsid omapoolseid soovitusi õppematerjali parandamiseks. Testijad leidsid, et kõige kasulikum harjutus oli teostuse plaani (*execution plan*) analüüsimine.

Töö autor õppis bakalaureusetööd koostades palju uut Microsoft SQL Server 2012 kohta ning oskab nüüd efektiivsemalt kasutada ja hallata andmebaasi serverit. Bakalaureusetöö võiks olla abiks neile, kes on eelnevalt tutvunud SQL keelega, kuid ei oska veel hallata andmebaasi ja analüüsida skripte. Loomulikult on võimalik seda tööd ka edasi arendada veel spetsiifilisemaks, kuid samas ka laiemaks, sest hetkel ei suutnud töö autor jagada kogu vajalikku materjali, mida algaja administraator peaks teadma. Lisaks võiks tutvuda ka kasutajate autoriseerimisega, andmete importimise ja eksportimisega, mitme serveri haldamisega.

Summary

The aim on this thesis titled “Administration of Microsoft SQL Server 2012. Learning Material” was to create a simple SQL Server administration tutorial in Estonian to benefit the learning and teaching of Microsoft SQL Server 2012. To achieve the goal the author examined the administration of Microsoft SQL Server 2012, identified the contents of the tutorial, created the learning material, carried out a small test with two people and analyzed the results.

Tutorial consisted of six chapters where author described different areas where beginner database administration must learn. These areas were recovery models, automating SQL Server Management, database maintenance, monitoring SQL Server, SQL Server trace and optimizing SQL queries.

According to the test participants the tutorial was rather easy to follow but lacked the number of exercises. Author took into these considerations and some recommendations for the learning material and improved the tutorial. Test participants found that the best exercise was about Actual Execution plan (Exercise 6) because it showed the importance of optimizing SQL queries and indexes. Additional exercises are available online in www.sandersql.eu.

For further development of the learning material is possible to study authentication, user groups, importing and exporting data. Furthermore there is a possibility to write additional exercises.

Kasutatud kirjandus

Kippar, J. (2011). *Jaagup Kippar Koostatud loengumaterjalid.* Allikas: Koostatud loengumaterjalid, SQL, päringud, tabelite sidumine, päästikprotsessid: http://minitorn.tlu.ee/~jaagup/kool/java/loeng/juht.html

Microsoft. (2012). *Microsoft Learning Course 10775A.* Allikas: Microsoft Learning Course 10775A: http://www.microsoft.com/learning/en-us/course.aspx?ID=10775A

Microsoft. (2014). *Microsoft*. Kasutamise kuupäev: 23. 04 2014. a., allikas SQL Server Editions: http://www.microsoft.com/en-us/server-cloud/products/sql-server-editions/#fbid=_i_P2yck66Y

Microsoft. (2014). *Microsoft Developer Network*. Kasutamise kuupäev: 28. 04 2014. a., allikas How To: Optimize SQL Indexes: http://msdn.microsoft.com/en-us/library/ff650692.aspx

Microsoft. (2014). *Microsoft Developer Network Editions and Components of SQL Server 2012*. Kasutamise kuupäev: 23. 04 2014. a., allikas Editions and Components of SQL Server 2012: http://msdn.microsoft.com/en-us/library/ms144275.aspx

Microsoft. (2014). *Microsoft Developer Network Features Supported by the Editions of SQL Server 2012*. Kasutamise kuupäev: 04. 05 2014. a., allikas Features Supported by the Editions of SQL Server 2012: http://msdn.microsoft.com/en-us/library/cc645993.aspx

Microsoft. (2014). *Microsoft Developer Network How To: Optimize SQL Queries*. Kasutamise kuupäev: 28. 04 2014. a., allikas How To: Optimize SQL Queries: http://msdn.microsoft.com/en-us/library/ff650689.aspx

Microsoft. (2014). *Microsoft Technet DBCC CHECKDB.* Allikas: Microsoft Technet DBCC CHECKDB: http://technet.microsoft.com/en-us/library/ms176064.aspx

Microsoft. (2014). *Microsoft Technet SQL Server Index Design Guide*. Allikas: SQL Server Index Design Guide: http://technet.microsoft.com/en-us/library/jj835095.aspx

Pilecki, M. (2008). *Microsoft*. Kasutamise kuupäev: 01. 05 2014. a., allikas TechNet Magazine: http://technet.microsoft.com/en-us/magazine/2007.11.sqlquery.aspx

Lisa 1 – Õppematerjal

Microsoft SQL Server 2012 Administreerimine.

Sander Kuusk

Tallinn 2014

Käesolev materjal:

- On mõeldud algajale SQL Serveri administreerijale;
- On loodud sellepärast, et eesti keeles leidub SQL Serveri administreerimise kohta; vähe informatsiooni
- Eeldab algteadmisi T-SQL keelest;
- Õpetab sammhaaval indeksite loomist, logide vaatamist, andmebaasi rikutuse otsimist, skriptide analüüsimist, varukoopiate tegemist ja serveri hetkeseisu kontrollimist;
- Kirjeldab indeksite olulisust ja loomist andmebaasi.

Selleks, et läbida antud õppematerjali, peab kasutaja arvutisse olema installeeritud *Microsoft SQL Server 2012 Express* versioon ja *AdventureWorks2012* andmebaas. Õppematerjali läbimiseks kulub umbes kaks tundi, eeldusel, et ettevalmistused on tehtud. Töö autor on ette valmistanud 6 harjutust, millega algajad andmebaasi administraatorid kokku puutuvad.

Sisukord

Indeksite loomise juhend

Kehvasti disainitud indeksid ja indeksite vähesus on peamised süsteemi kitsaskohtade allikaks. Efektiivsete indeksite disainimine on olulisim lüli, et saavutada hea andmebaasi ja rakenduste kiirus. Antud SQL Serveri indeksite disainimise juhend sisaldab informatsiooni ja parimaid praktikaid kuidas aidata luua efektiivseid indekseid, mis täidaksid rakenduse vajadused. (Microsoft, 2014)

Indeks on struktuur, mis on seotud tabeli või *view*'ga. Indeks sisaldab võtmeid, mis on loodud ühest või mitmest veerust tabelis või *view*'s. need võtmed salvestatakse struktuuri, mis võimaldab SQL Serveril leida üles vastavad read, mis on seotud nende võtmete väärtustega kiiresti ja efektiivselt. (Microsoft, 2014)

Valik õigetest indeksitest andmebaasis on keerukas tegevus, mis balansseerib päringu kiiruse ja uuendamise kulukuse vahel. Kitsad indeksid või indeksid, mis on väheste veergude peale üles ehitatud võtavad vähem kettaruumi ja on kergem hallata. Laiad indeksid katavad, aga rohkem päringuid. Sellepärast, tulebki eksperimenteerida mitmete erinevate disainidega, et leida kõige efektiivsem indeks. Indekseid on võimalik lisada, modifitseerida ja kustutada ilma, et see mõjutaks andmebaasi skeeme või rakenduste disaine. Seetõttu ei tohiks kõhelda, et eksperimenteerida erinevate indeksitega. (Microsoft, 2014)

Päringu optimeerija (*Query optimizer*) valib üldjuhul kõige efektiivsema indeksi, mida kasutada päringutes. Üleüldine indeksite disain võiks võimaldada mitmekesiseid indekseid, mille seast saaks päringu optimeerija ise valida. See vähendab analüüsi aega ja toodab hea jõudluse üle mitmete erinevate situatsioonide. Kui on soov näha, mis indekseid päring kastab, siis selleks tuleks *SQL Server Management Studio* alt valida *Include Actual Execution Plan*. (Microsoft, 2014)

Kunagi ei tasu mõelda nii, et indeksite kasutamine võrdub hea päringu jõudlusega, sest ebaõige indeksi kasutamine päringus võib jõudlust vähendada. Selleks ongi päringu optimeerija, mis valib indeksi või kombinatsiooni indeksitest, et suurendada jõudlust ja vältida indekseid, mis vähendavad jõudlust. (Microsoft, 2014)

Indeksite disainimise tasuks järgida vastavat plaani: (Microsoft, 2014)

1. Aru saada andmebaasi tunnustest.
2. Aru saada korduvalt kasutatud päringutest.
3. Aru saada veergude tunnustest, mida kasutatakse päringus
4. Välja selgitada, millised on indeksi loomise võimalused, et suurendada päringu jõudlust.
5. Otsustama kõige optimaalsema indeksi varundamise koha.

Kogemustega andmebaasi administraatorid oskavad disainida häid indekseid, kuid see ülesanne on väga keeruline, aja nõudlik ja vigade rohke, kui tegemist on veidi keerukama

andmebaasiga. Aru saada andmebaasi, päringu ja veergude tunnustest aitavad algajal administraatoril disainida optimaalsemaid indekseid.

Aru saamine indeksitest andmebaasi tasandil. (Microsoft, 2014)

- Suur number indekseid tabelis mõjutavad INSERT, UPDATE, DELETE ja MERGE lausete jõudlust, sest kõik indeksid peavad kohandama ennast korrektselt andmete muutmisele. Näiteks kui veergu, mida kasutavad mitmed indeksid, uuendatakse UPDATE lausega, mis modifitseerib veerus leiduvaid andmeid, siis iga indeks, mis sisaldab seda veergu peavad ka uuendama.
    - Väldi liigsete indeksite loomist ja hoia indekseid kitsastena, see tähendab kasuta võimalikult vähe veerge.
    - Kasuta indekseid, et suurendada jõudlust tabelites, kus on vähe uuendamise vajadust, kuid sisaldavad palju andmeid. Suur number indekseid võivad aidata kaasa lausetele, mis ei sisalda andmete modifitseerimist näiteks SELECT.
- Väikeste tabelite indekseerimine ei pruugi olla optimaalne kuna päringu optimeerijal läheb kauem aega, et teha indeksite otsimine. Sama kulu eest oleks võimalik teha ka lihtne tabeli skaneerimine. Seega indekseid, mis on väikestes tabelites ei pruugita üldse kasutada.
- Indekseeritud *view*'d võivad tagada suure jõudluse kasvu, kui viimane sisaldab liitmisi (*aggregation*), tabelite ühendumisi või nende kombinatsioone.
- Kasuta *Database Engine Tuning Advisor*'it, et analüüsida oma andmebaasi ja luua indeksite soovitusi.

Aru saamine indeksitest päringu tasandil. (Microsoft, 2014)

- Loo *nonclustered* indekseid veergudel, mida tihti kasutatakse tabeli ühenduste (*join*) tegemisel. Liiga palju tehtud indekseid mõjuvad laastavalt nii kettaruumile kui ka hoolduse jõudlusele.
- Indeksite katmine võib suurendada päringu jõudlust, kuna kõik andmed, mida päritakse on juba indekseeritud.
- Kirjuta päringuid, mis lisavad või modifitseerivad nii mitu rida kui võimalik ühes lauses. Kasutades vaid üht lauset, saab optimeeritud indeksite haldust ära kasutada
- Hinda päringu tüüpi ja kuidas kasutatakse veerge päringus.

Aru saamine indeksitest veeru tasandil. (Microsoft, 2014)

- Hoia *clustered* indeksite võtme pikkus lühike.
- Veerud, mis on *ntext*, *text*, *image*, *varchar(max)*, *nvarchar(max)* ja *varbinary(max)* andmetüüpi ei saa olla indekseeritud võtmeveergudeks. Samas *varchar(max)*, *nvarchar(max)*, *varbinary(max)* ja *xml* andmetüüpi veerud võivad osaleda nonclustered indeksi loomisel.
- XML andmetüüp saab olla võtmeveerg vaid XML indeksile.

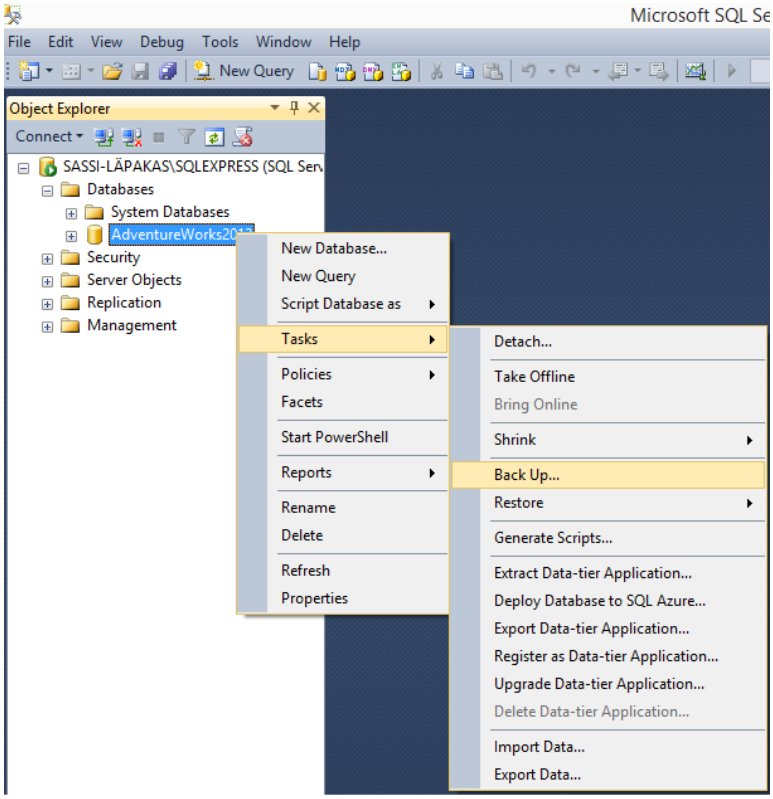
- Uuri veergude unikaalsust. Unikaalne indeks sisaldab rohkem informatsiooni mida päringu optimeerija saaks kasutada võrreldes *nonunique* indeksiga.
- Uuri andmete jaotust veerus. Sageli päring, mis võtab kaua aega, võib olla põhjustatud ühendumistest veergude vahel. See on fundamentaalne probleem ja reeglina ei saa lahendada ilma suurema uurimiseta.
- Hinda filtreeritud indeksite kasutamist veergudel, milledel on hästi defineeritud alamhulgad (*subset*).
- Hinda veergude järjestust, kui indeks sisaldab mitut veergu.
- Kaalutle arvutatud (*computed*) veergude indekseerimist.

Kui on välja valitud õige indeks päringu jaoks tasuks üle vaadata indeksite tunnused, mis sisaldavad järgmisi (Microsoft, 2014)

- *Clustered* versus *nonclustered*
- *Unique* versus *nonunique*
- Üks veerg versus mitu veergu
- Kasvav või kahanev järjestus indekseeritud veerule.
- Terve tabeli otsing versus *nonclustered* indeks.

Harjutus 1 Varukoopia tegemine AdventureWorks2012 andmebaasist

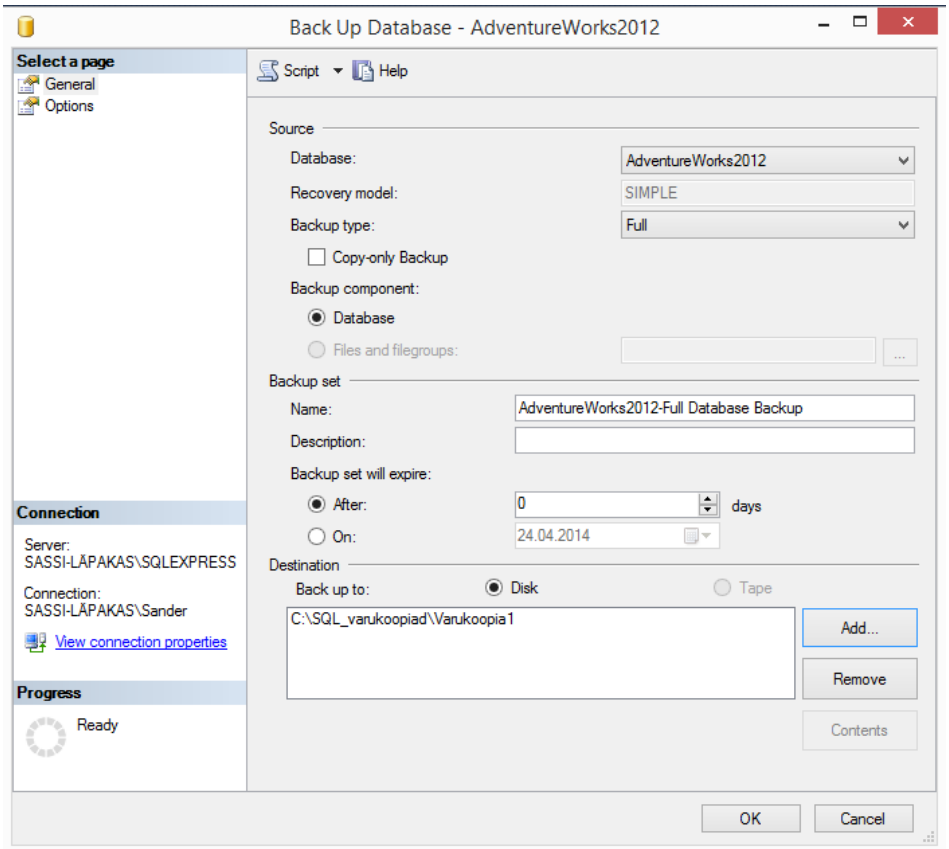
1. Kasutades *Windows Explorer*'it loo uus kaust C:\SQL_varukoopiad.
2. Avades *SQL Server Management Studio* leida üles *AdventureWorks2012* andmebaas *Object Explorer*'ist.
3. Parem hiireklikk *AdventureWorks2012* andmebaasi peal, edasi *Tasks* ja *Back up*.



Joonis 6 Varukoopia tegemine

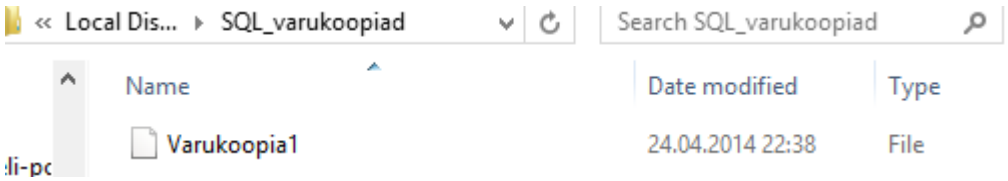
4. Eemalda varukoopia sihtpunkt (*Destination*) ja lisa sinna uus.

5. Asukohaks tuleks määrata just loodud kaust C:\SQL_varukoopiad ja nimeks Varukoopia1 ja vajutada OK.



Joonis 7 Varukoopia salvestamine

6. Varukoopia asub kaustas C:\SQL_varukoopiad.

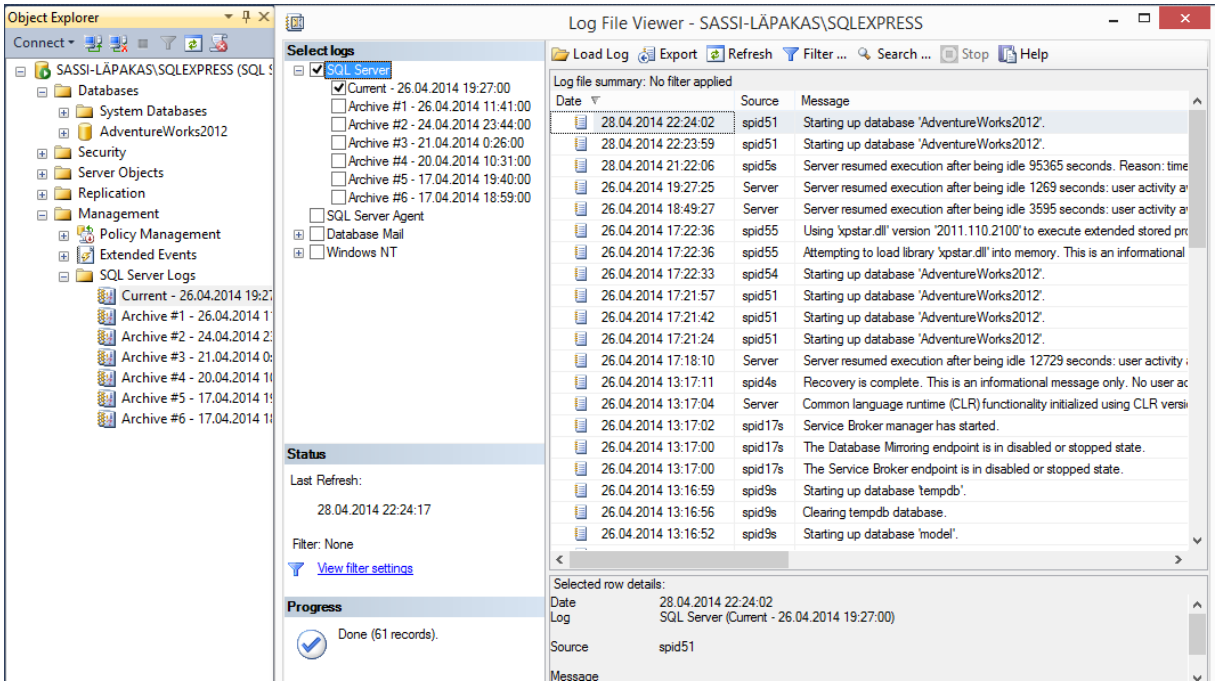


Joonis 8 *AdventureWorks 2012* Varukoopia

Mis lisavõimaluse annab *Copy-only Backup* juurde tehes varukoopiat?

Harjutus 2 *SQL Server Logs* kasutamine

Andmebaasi administraatoril tuleb aeg ajalt vaadata SQL Serveri logisid, et vaadata üle kas kõik toimib korrektselt ja ega veateateid ei tule. Microsoft on teinud logide vaatamise väga lihtsaks *SQL Server Management Studios*. Selleks peab valima *Object Explorer*'ist *Current SQL Server Logs*, mis asub *Management* valiku all.

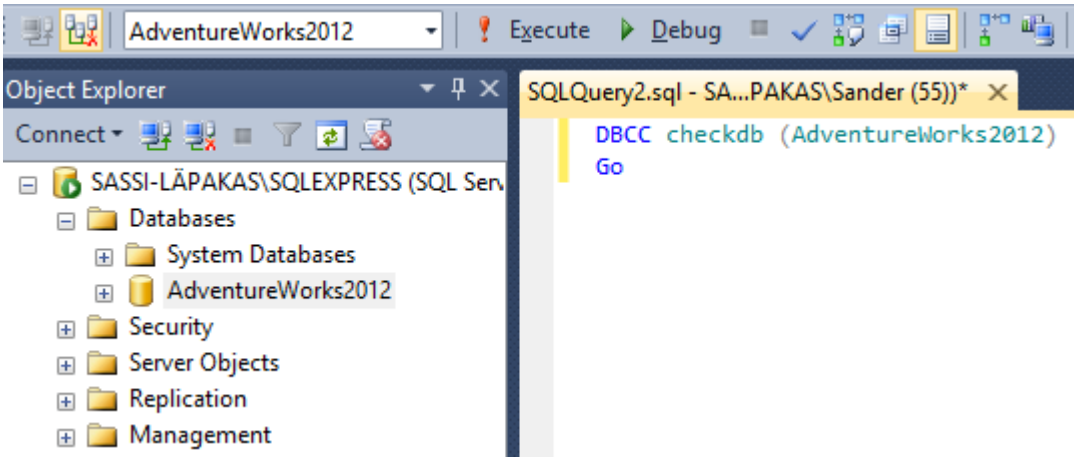


Joonis 9 SQL Server logide kasutamine

Siin saab administraator veenduda, et kõik protsessid on edukalt lõpule viidud näiteks varukoopia tegemine.

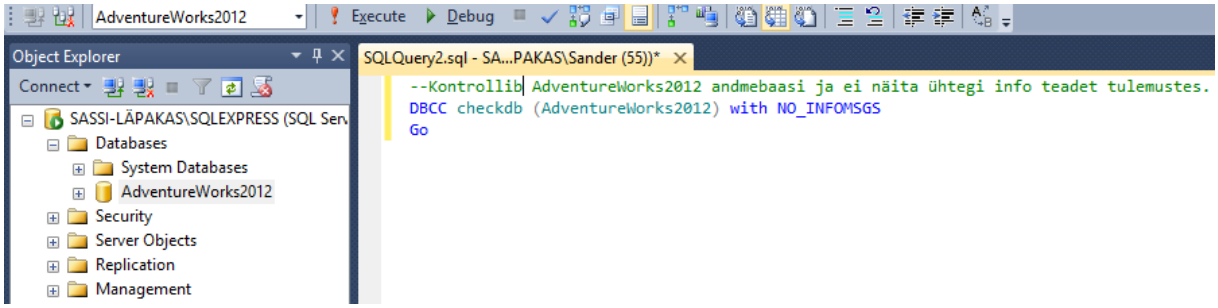
Harjutus 3 Andmebaasi rikutuse kontrollimine kasutades käsklust DBCC CHECKDB

Andmebaasi rikutuse kontrollimine on võrdlemisi lihtne, selleks läheb vaja vaid paari koodirida. Joonis 10 *DBCC CHECKDB* kasutamine



Joonis 10 *DBCC CHECKDB* kasutamine

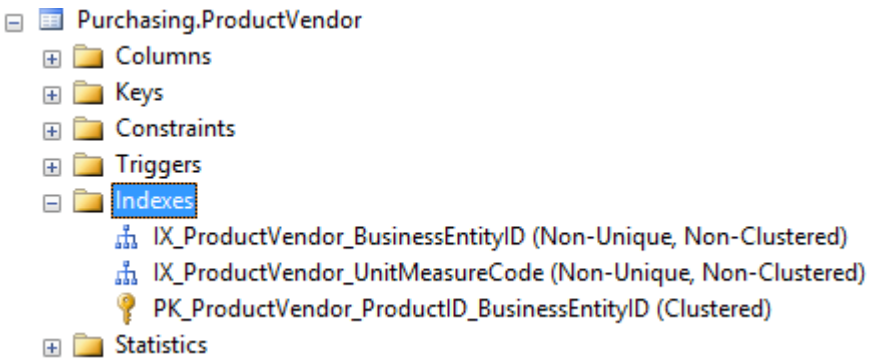
Antud koodirida kontrollib AdventureWorks2012 andmebaasi. Lisaks tagastab see koodirida mitmeid info teateid tulemustes, mida võib olla ebamugav lugeda, sest neid on väga palju. Selleks võib lisada koodi reale *with NO_INFOMSGS*. Joonis 11 *DBCC CHECKDB with NO_INFOMSGS* kasutamine



Joonis 11 *DBCC CHECKDB with NO_INFOMSGS* kasutamine

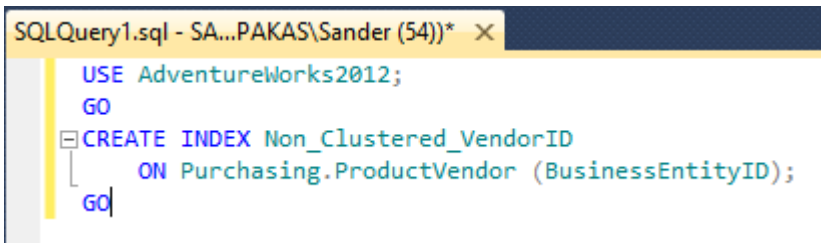
Harjutus 4 *Non-Clustered* Indeksi loomine

Järgnevalt on tarvis luua *Non-Clustered* indeks *AdventureWorks2012* andmebaasis asuvale *Purchasing.ProductVendor* tabeli *VendorID* veerule. Selleks tasub eelnevalt üle vaadata mis indeksid antud tabelis on.



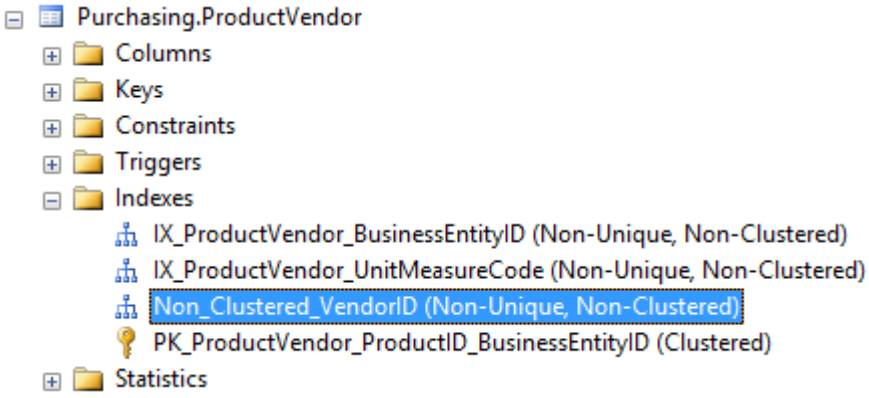
Joonis 12 Purchasing.ProductVendor tabeli indeksid

Edasi tuleks lisada uus indeks, mis oleks *Non-Clustered* ja viitaks BusinessEntityID peale



Joonis 13 Non-Clustered indeksi lisamine

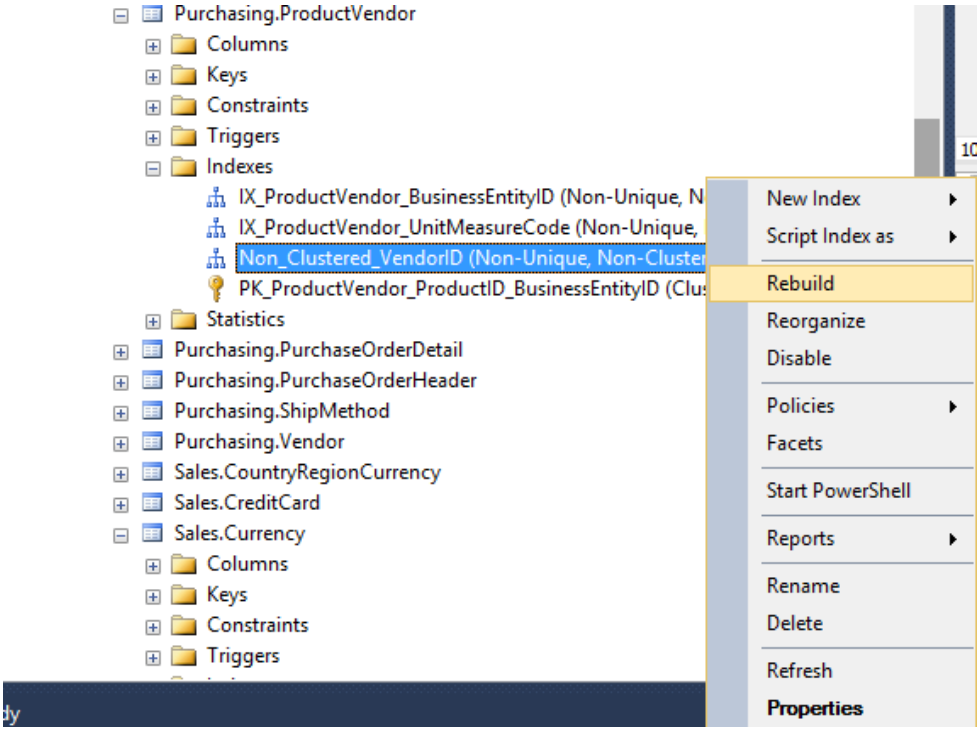
Kontrollime, kas vastav indeks lisati Purchasing.ProductVendor tabelile juurde.



Joonis 14 Non-Clustered indeks

Harjutus 5 Indeksi uuesti üles ehitamine (*Rebuild*) ja ümber korraldamine (*Reorganize*)

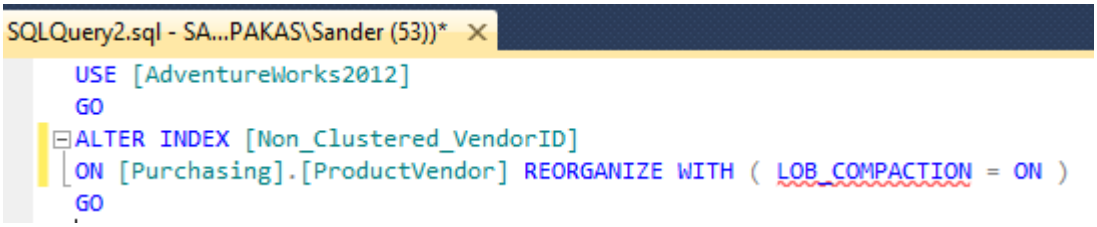
Töö autor kasutab indeksi uuesti üles ehitamiseks sisseehitatud graafilist lahendust, kuid indeksi ümberkorraldamise puhul skripti. Selleks, et indeks uuesti üles ehitada, tuleb see selekteerida ning valida alammenüüst *Rebuild*. Joonis 15 Indeksi uuesti ümber ehitamine



Joonis 15 Indeksi uuesti ümber ehitamine

Avanenud aknas on võimalik veel andmed üle kontrollida. Eeldusel, et valitud sai õige indeks võib vajutada avanenud aknas OK nuppu. Viimane paneb käima indeksi uuesti ümber ehitamise protsessi.

Järgnevalt tuleb luua koodirida indeksi ümber korraldamise kohta ning see käivitada, kasutades *Microsoft SQL Server Management Studio*'t. Joonis 16 Indeksi ümber korraldamise koodiread

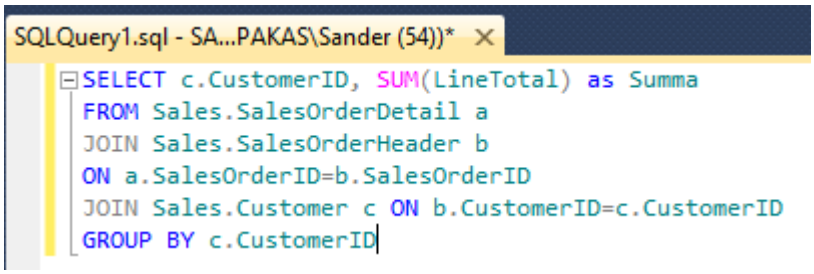


Joonis 16 Indeksi ümber korraldamise koodiread

LOB_COMPILATION tähendab seda, et kõik lehed, mis sisaldavad suuri andme objekte (*image, text, ntext, varchar(max), nvarchar(max)* ja *xml*) tihendatakse (*Compacted*).

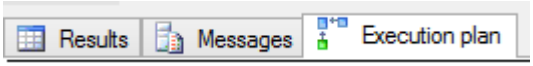
Harjutus 6 *Execution Plan* analüüsimine.

Järgnevalt koostame lihtsa päringu, vaatame teostuse plaani ja otsime võimalusi jõudluse parandamiseks. Käivitades päringut tuleks vaadata, et *Actual Execution Plan* valik oleks sisse lülitatud *SQL Server Management Studio*'s (lühivalik Ctrl + M). Järgnev skript kalkuleerib iga kliendi tellimuste summa. (Microsoft, 2014)



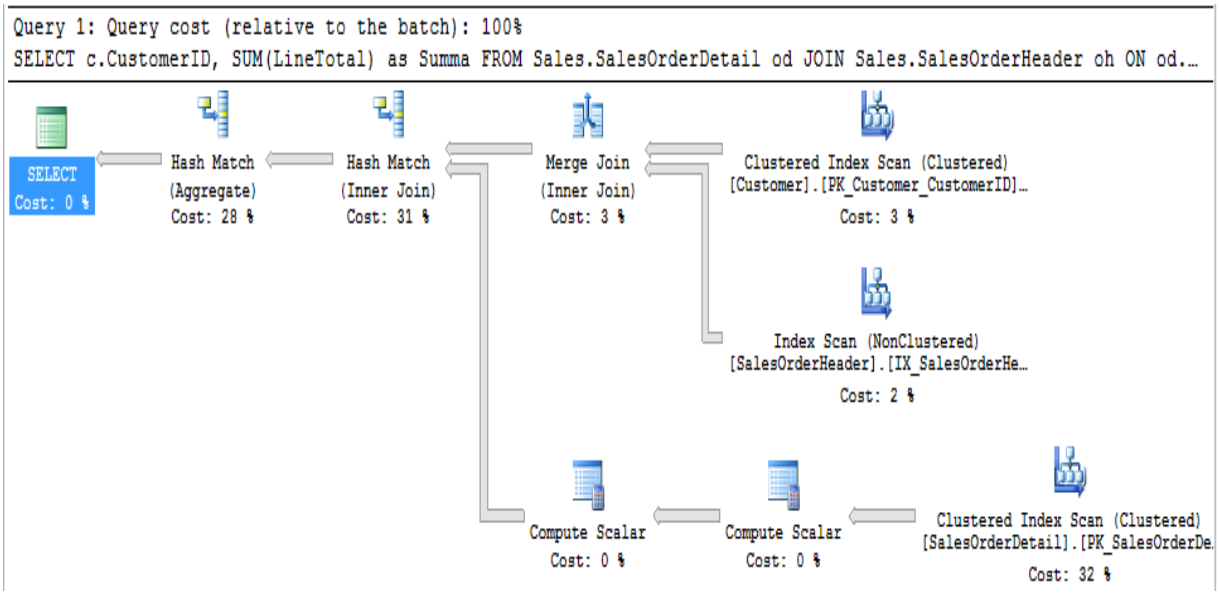
Joonis 17 skript analüüsimiseks

Käivitades vastava skripti tekib peale tulemuste ja sõnumite ka teostuse plaani sakk *SQL Server Management Studio*'sse. Joonis 18 *Execution Plan*



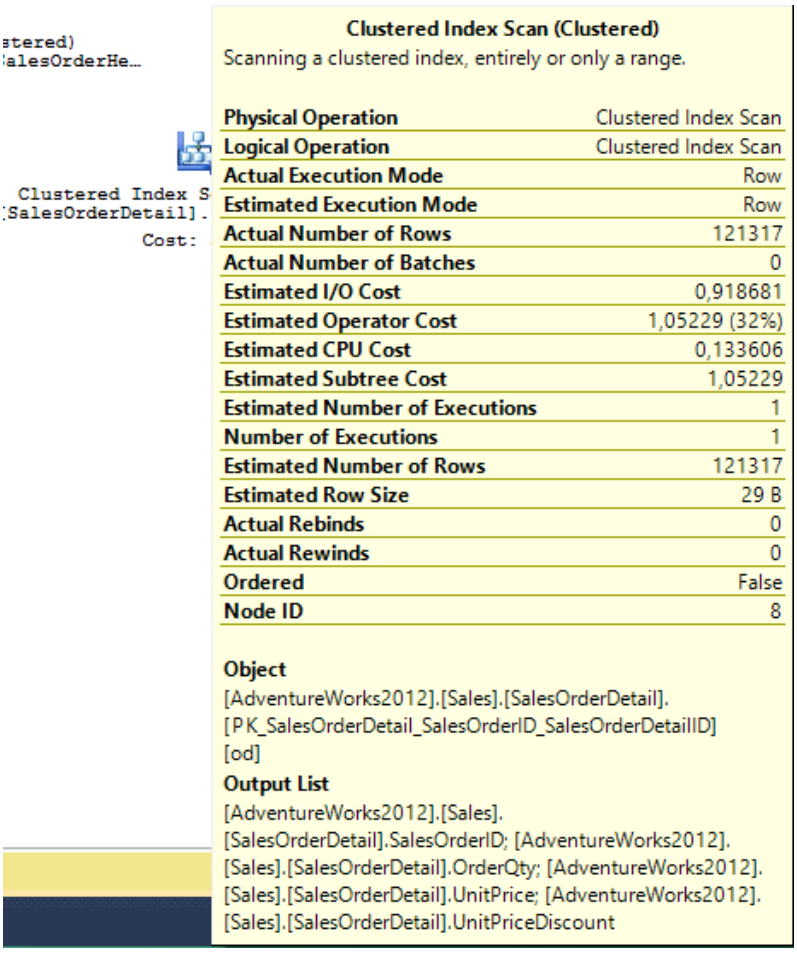
Joonis 18 *Execution Plan*

Vaadates teostuse plaani, siis on võimalik näha, kuidas andmebaasi mootori töötleb päringut ja valmistab ette tulemust. Graafilist teostuse plaani on võimalik lugeda nii ülevalt alla kui paremalt vasakule. Iga ikoon tähistab nii loogilist kui ka füüsilist operatsiooni ja nooled näitavad andmete liikumist nende vahel. Mida laiem on joon seda rohkem ridu töödeldi.



Joonis 19 ***Actual Execution Plan***

Kui panna hiire osuti üle mõne ikooni, siis näidatakse antud operatsiooni detaile.

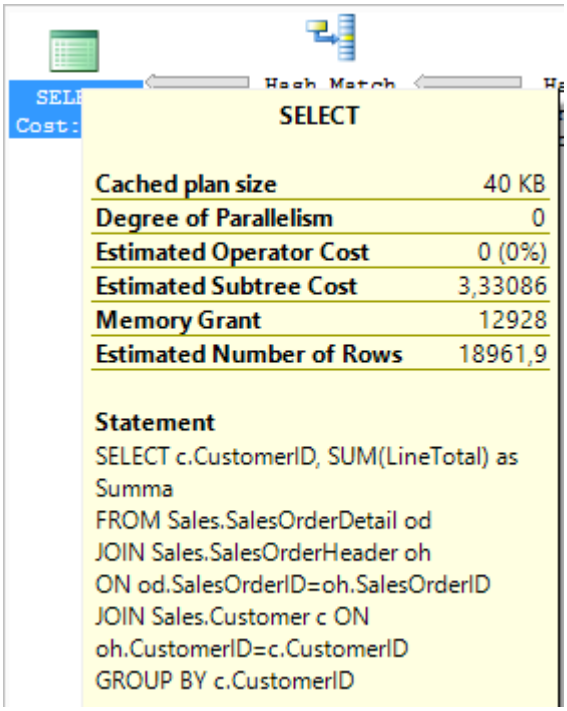


Joonis 20 ***Actual Execution Plan*****'i detailid**

Järgnevalt tasuks analüüsida igat operaatorit, et aru saada mis järjekorras midagi tehakse. (Microsoft, 2014)

1. Andmebaasi mootor teeb *Clustered Index Scan* operatsiooni *Sales.Customer* tabeli pihta ja tagastab *CustomerID* veeru igale reale selles tabelis.
2. Järgnevalt sooritatakse *Index Scan* ühele indeksile, mis asub *Sales.SalesOrderHeader* tabelis. See indeks *ColumnID* veerus, kuid kaudselt (*implicitly*) on see seotud ka *SalesOrderID* veeruga (*Clustered Key*). Skaneerimise käigus tagastatakse mõlema veeru väärtused.
3. Mõlema skaneerimise tulemused ühendatakse (*Merge Join*) läbi *CustomerID* veeru. See on üks kolmest võimalikust loogilise ühendamise võimalusest. See on kiire, kuid nõuab mõlema sisendi sorteerimist ühendatud veerule. Praegusel juhtumil on mõlemad skaneerimise operatsioonid tagastanud sorteeritud *CustomerID* read see tähendab, et eraldi sorteerimis operatsiooni ei ole vaja kasutada.
4. Järgnevalt skaneerib andmebaasimootor *Sales.SalesOrderDetail* tabelis olevat rühmitatud indeksit (*Clustered index*). Andmebaasi mootorile tagastatakse nelja veeru (*SalesOrderID*, *OrderQty*, *UnitPrice* ja *UnitPriceDiscount*) väärtused. Andmebaasi mootor eeldas, et päring tagastab 121317 rida ning tõesti see number ridu tõesti tagastati. Joonis 20 *Actual Execution Plan'i* detailid.
5. *Clustered* indeksi skaneerimisel tagastatud read edastatakse esimesse *Compute Scalar* operatsiooni niimoodi, et igale reale saab arvutada summa (*Line Total*), mis põhineb *OrderQty*, *UnitPrice* ja *UnitPriceDiscount* veergudel.
6. Teine *Compute Scalar* operatsioon lisab ISNULL funktsiooni tagastatud väärtustele. See täiendab summa veeru kalkulatsiooni ning edastab selle koos *SalesOrderID* veeruga järgmisele operatsioonile.
7. Tulemus, mis saadi ühendamise (*Merge Join*) teel kolmandas sammus ühendatakse omakorda tulemusega, mis saadi *Compute Scalar* operatsioonist kuuendas sammus.
8. Järgmine *Hash Match* operaator rakendatakse ridadele, mis tagastati ühendamisel (*Merge Join*) Sinna arvutatakse juurde SUM, mis saadakse summa (*LineTotal*) veerust.
9. Viimaseks on SELECT lause, mis ei ole ei füüsiline ega ka loogiline operatsioon, vaid näitab päringu üleüldisi tulemusi ja kulu

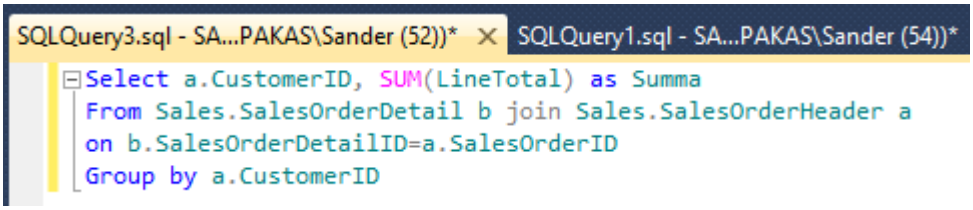
Töö autori sülearvuti peal näitab selle tulemuse plaan päringu eelkalkulatsiooni kulu 3,33086. (Joonis 21 *Select* lause detailid)



Joonis 21 *Select* lause detailid

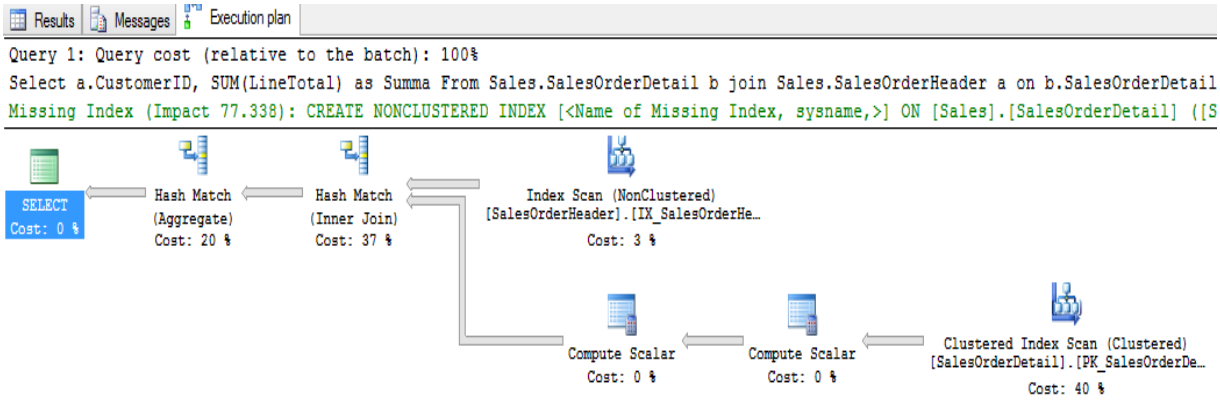
Vaadates üleüldist plaani, siis 3 operatsiooni on kõige kulukamad: *Cluster* indeks skaneerimine *Sales.SalesOrderDetail* tabeli peal ja kaks *Hash Match* operatsiooni.

Järgnevalt üritab tööautor elimineerida kaks operaatorit korraga, et parandada päringu kiirust. Kuna ainuke veerg, mida *Sales.Customer* tabelist vajame on *CustomerID* ja see on ka võõr võti (*Foreign Key*) *Sales.SalesOrderHeader* tabelis, siis võib selle elimineerida *Customer* tabeli päringust. Antud tegu ei tohiks ühtegi loogilist muutust päringusse sisse tuua. Joonis 22 Muudetud skript (Microsoft, 2014)



Joonis 22 Muudetud skript

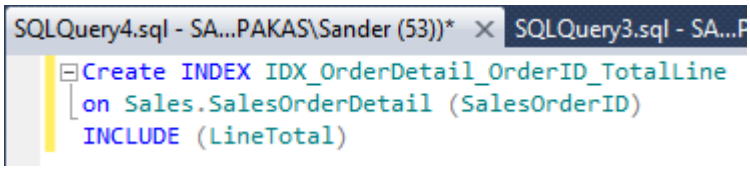
Loomulikult kajastuvad muudatused ka teostuse plaanis (*Execution Plan*). Joonis 23 Uus *execution plan*



Joonis 23 Uus *execution plan*

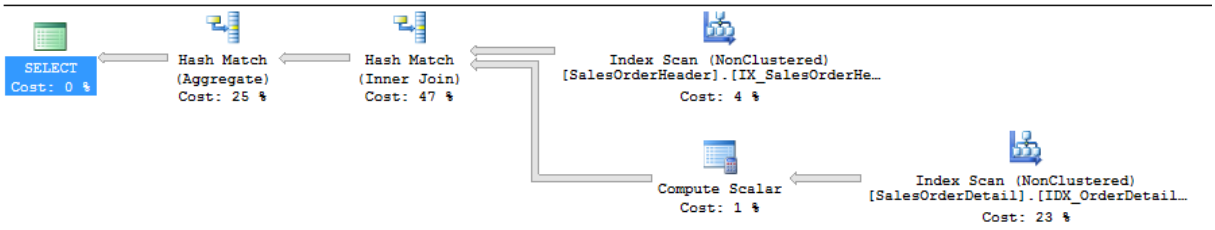
Kaks operaatorit on täielikult elimineeritu, *Clustered* indeksi skaneerimine *Customer* tabelil ja ühendamine (*Merge Join*) *Customer*'i ja *SalesOrderHeader*'i vahel. *Merge Join* on oluliselt efektiivsem kui *Hash Match,* kuid viimase jaoks peavad olema read mõlematest tabelitest (*SalesOrderHeader* ja *SalesOrderDetail*) sorteeritud *SalesOrderID* järgi. Selle saavutamiseks pidi päringu optimeerija skaneerima *Clustered* indekseid *SalesOrderHeader* tabelis, mitte kasutama *Non-Clustered* indeksite skaneerimist, mis samas oleks olnud „odavam" mõeldes I/O peale. See on hea näide, kuidas päringu optimeerija töötab. Tänu füüsilistele ühenduste muudatustele tekkis vähem kulu, siis päringu optimeerija valis kõige madalama eelkalkulatsiooniga teostuse kuluga plaani. Töö autori sülearvutil kulus selle päringu peale 2,63228 sekundit, mis on ligikaudu 21% parem aeg võrreldes eelmise skriptiga.

Järgnevalt tasuks vaadata järgmist kõige kallimat operatsiooni (40%), milleks on *Clustered Index Scan SalesOrderHeader* tabelil. Kuna vajalik on vaid kaks veergu antud tabelist, siis on parim valik koostada *Non-Clustered* indeks, mis sisaldab kahte veergu. Targem on välja vahetada terve tabeli skaneerimine palju väiksema *non-clustered* indeksi vastu. Selleks koostame indeksi nimega *IDX_OrderDetail_OrderID_TotalLine*. Joonis 24 Uue *non-clustered* indeksi loomine (Microsoft, 2014)



Joonis 24 Uue *non-clustered* indeksi loomine

Kui uus indeks on loodud, siis võib päringu uuesti tööle panna ja vaadata siis *Execution* plaani



Joonis 25 *Execution Plan* peale uue indeksi lisamist

Clustered indeksi skaneerimine on asendatud *NonClustered* indeksi skaneerimisega, mis on oluliselt väiksema I/O kuluga. Samuti elimineeriti üks *Compute Scalar* operatsioon kuna just loodud indeks juba arvutab välja summa (*LineTotal*) veeru. Nüüd on *execution* plaani kulu 2,0483, mis on omakorda eelmisest päringust ligikaudu 22% kiirem. See teeb kokku umbes 38,7% kiirema päringu võrreldes esialgse skriptiga.