Održavanje baze podataka

Miličević, Milka

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Milka Miličević

ODRŽAVANJE BAZE PODATAKA

Diplomski rad

Voditelj rada:
prof. dr. sc. Robert Manger

Zagreb, rujan, 2019

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvaljujem se mentoru prof. dr. sc. Robertu Mangeru na pomoći i svim sugestijama vezanim uz ovaj rad, te svim onima koji su na bilo koji način doprinijeli nastanku ovog rada. Najveće hvala mojoj obitelji na razumijevanju i nesebičnoj podršci tijekom studija.

Sadržaj

Uvod

Baza podataka je skup međusobno povezanih podataka, pohranjenih u vanjskoj memoriji računala koji su istovremeno dostupni raznim korisnicima i aplikacijskim programima. Ubacivanje, promjena, brisanje i čitanje podataka obavlja se posredstvom posebnog softvera, takozvanog sustava za upravljanje bazom podataka (DBMS-a). Osoba koja je odgovorna za osiguravanje stalne operativne funkcionalnosti i učinkovitosti baza podataka i aplikacija koje pristupaju tim bazama podataka je administrator baze podataka (DBA).

U ovom diplomskom radu obradit ću temu pod nazivom održavanje baze podataka. Diplomski rad je podijeljen na nekoliko poglavlja. U prvom poglavlju bavimo se poslovima koje obavlja administrator baze podataka, a ti poslovi su: oblikovanje baze podataka, praćenje i podešavanje performansi, osiguravanje dostupnosti baze korisnicima, čuvanje sigurnosti podataka, stvaranje rezervnih kopija baze podataka, oporavak baze u slučaju njezinog oštećenja i osiguravanje integriteta podataka.

U idućem poglavlju opisujemo DBA alate. DBA alati smanjuju količinu vremena, truda i ljudske pogreške u održavanju učinkovitih sustava i baza podataka. Takvi alati olakšavaju administrativno opterećenje i smanjuju mogućnost pogreške. U tom poglavlju se daje pregled kategorija i vrsta proizvoda dostupnih DBA-u kao pomoć u upravljanju i održavanju baza podataka.

Studijski primjer baze podataka čini zadnje poglavlje ovog rada, gdje smo kao DBMS koristili Microsoft SQL Server. Tu je prikazana sama izgradnja baze padataka i njezinih objekata kao što su: tablice, pogledi, indeksi, uskladištene procedure, funkcije i okidači. Kroz izrade shema, prijavni naloga, korisnički računa, uloga i kriptiranje podatka prikazana je organizacija i sigurnost baze podataka. Nakon toga su prikazani načini izrade rezervnih kopija, povrata baze podatka, kopiranja baze podataka i plana održavanja. Tu je još opisan i SQL Server Agent koji je zadužen za automatizaciju administrativnih poslova u SQL Serveru.

Za izradu Chenovog dijagrama koristili smo Visual Paradigm[5], a Microsoft SQL Server[6] smo koristili kao sustav za upravljanje bazom podataka.

Poglavlje 1

Poslovi administratora baze podataka

Administrator baze podataka (DBA) je informacijski stručnjak odgovoran za osiguravanje stalne operativne funkcionalnosti i učinkovitosti baza podataka i aplikacija koje pristupaju tim bazama podataka.

On mora biti sposoban obavljati mnoge zadatke kako bi osigurao korisne, upotrebljive, dostupne i točne podatke iz baze podataka. Ti zadaci uključuju oblikovanje, praćenje i podešavanje performansi, osiguravanje dostupnosti baze korisnicima, autoriziranje sigurnosti, stvaranje rezervnih kopija i oporavak baze u slučaju njezinog oštećenja, osiguravanje integriteta podataka, zapravo sve što je povezano s bazama podataka. U nastavku rada razmotriti ćemo svaku od ovih tema.

1.1 Oblikovanje baze podataka

Prvi zadatak na koji većina ljudi pomisli kada je riječ o administratoru baze podataka je sposobnost stvaranja dobro osmišljenih baza podataka. Administratori baze podataka moraju razumjeti i poštivati prakse relacijskog oblikovanja kako bi pravilno oblikovali i stvorili relacijske baze podataka. Oni moraju razumjeti i relacijsku teoriju i specifičnost implementacije RDBMS-a (*relational database management system*) koja se koristi za stvaranje baze podataka. Oblikovanje baze podataka zahtijeva dobro razumijevanje konceptualnih i logičkih tehnika za modeliranje podataka. Sposobnost stvaranja i tumačenja dijagrama entitetskih odnosa nužna je za projektiranje relacijske baze podataka.

Nadalje, DBA mora biti u stanju transformirati logički model podataka u fizičku implementaciju baze podataka. DBA mora osigurati da oblikovanje i implementacija baze podataka omoguće korisnu bazu podataka za aplikaciju i klijente koji će je koristiti.

Doista, oblikovanje baze podataka je značajna vještina koju DBA posjeduje. Međutim, posao DBA-a često je nerazmjerno povezan s oblikovanje baze podataka. Iako je oblikovanje optimalnih baza podataka važno, to je relativno mali dio posla koje DBA obavlja. DBA

će najvjerojatnije potrošiti više vremena na upravljanje i podešavanje baze podataka nego na oblikovanje i izgradnju baza podataka.

Međutim, ni u kom slučaju se ne smije protumačiti da oblikovanje baze podataka nije važno. Loša relacijska konstrukcija može rezultirati lošijom performansom, bazom podataka koja ne zadovoljava potrebe organizacije, te potencijalno netočnim podacima.

1.2 Praćenje i podešavanje performansi

Druga uloga koja je usko povezana s DBA je praćenje i podešavanje performansi. Stopa po kojoj DBMS (database management system) zadovoljava potražnju za informacijama može se nazvati performansom baze podataka. Ali, to nije tako jednostavno. Pet faktora utječe na performanse baze podataka, a to su:

- *Radno opterećenje.* Radno opterećenje koje se traži od DBMS-a definira zahtjev. To je kombinacija online transakcija, skupnih poslova, ad hoc upita, skladištenja podataka i analitičkih upita, te naredbi usmjerenih kroz sustav u bilo kojem trenutku. Radno opterećenje može se drastično mijenjati iz dana u dan, iz sata u sat, iz minute u minutu, pa čak i iz sekunde u sekundu. Ponekad se opterećenje može predvidjeti (kao što je obrada plaće na kraju mjeseca ili vrlo lagan pristup nakon 19:30 sati kada je većina korisnika otišla za taj dan), ali u drugim slučajevima je nepredvidljivo. Cjelokupno radno opterećenje ima veliki utjecaj na performanse baze podataka.

- *Propusnost.* Propusnost definira ukupnu sposobnost računalnog hardvera i softvera za obradu. To je kombinacija I/O brzine, brzine procesora, paralelnih mogućnosti stroja i učinkovitosti operativnog sustava i softvera.

- *Resursi.* Hardverski i softverski alati koji su na raspolaganju sustavu poznati su kao resursi sustava. Primjeri uključuju jezgru baze podataka, prostor na disku, kontrolere predmemorije i mikrokod.

- *Optimizacija.* Svi tipovi sustava mogu se optimizirati, ali relacijski upiti su jedinstveni u toj optimizaciji koja se primarno ostvaruje unutar DBMS-a. Međutim, postoje mnogi drugi čimbenici koje treba optimizirati (SQL formulacija, parametri baze podataka, učinkovito programiranje i tako dalje) kako bi se omogućilo optimizatoru baze podataka da stvori najučinkovitije pristupne staze.

- *Sukob.* Kada je potražnja (opterećenje) za određeni resurs visoka, može doći do sukoba. Sukob je stanje u kojem dvije ili više komponenti radnog opterećenja pokušavaju koristiti jedan resurs na konfliktan način (na primjer, dvostruko ažuriranje istog dijela podataka). Kako se sukob povećava, propusnost se smanjuje.

Stoga se performansa baze podataka može definirati kao optimizacija korištenja resursa kako bi se povećala propusnost i smanjili sukobi, čime se omogućuje obrada najvećeg mogućeg radnog opterećenja.

Kada se naiđe na probleme s performansama u aplikaciji koja koristi bazu podataka, DBA je obično prvi koji se poziva na rješavanje problema. Aplikacije redovito komuniciraju s drugim aplikacijama, sustavima i komponentama IT infrastrukture. Učinkovita strategija praćenja i podešavanja performansi zahtijeva ne samo ekspertizu DBMS-a, nego i znanja izvan okvira administracije baze podataka. Mnoge zadatke upravljanja performansama moraju dijeliti DBA i drugi tehničari. Drugim riječima, rukovanje problemima performansi je doista pothvat na razini cijelog poduzeća.

DBA mora biti oprezan u sustavu praćenja, baza podataka i performansama aplikacija. Koliko god je to moguće to treba postići automatiziranim softverom i skriptama. Sistemske tablice s upitima i izrada upozorenja na temelju pragova mogu se koristiti za proaktivno prepoznavanje problema. Upozorenja mogu biti postavljena za slanje e-maila DBA-u kada podaci o performansama nisu unutar prihvaćenih granica.

Brojni zadaci i sposobnosti potrebni su DBA-u kako bi osigurao učinkovit pristup bazama podataka. Neke od tih sposobnosti uključuju izgradnju odgovarajućih indeksa, navođenje dovoljno velikih međuspremnika i privremenih spremnika, usklađivanje implementacije baze podataka s IT infrastrukturom, stalno praćenje baza podataka i aplikacija, reorganizaciju baza podataka i prilagođavanje poslovnim promjenama - više korisnika, više podataka, dodatna obrada, te promjenjivi zahtjevi i propisi.

1.3 Osiguravanje dostupnosti

Dostupnost podataka i baza podataka često je usko povezana s performansama, ali to je zapravo posebna briga. Naravno, ako je DBMS odsutan, performanse će biti loše jer nema pristupa podacima. Ali osiguravanje dostupnosti baze podataka je višestruki proces.

Prva komponenta dostupnosti je držati DBMS pokrenutim. Pažljivo praćenje i automatizirana upozorenja mogu se koristiti za upozoravanje na ispade DBMS-a i poziv na korektivne mjere.

Pojedinačne baze podataka također moraju biti održavane tako da su podaci sadržani u njima dostupni kad god je to potrebno. Na taj način DBA mora oblikovati bazu podataka tako da se može održavati uz minimalne poremećaje, ali i da pomogne u oblikovanju aplikacija kako bi se minimizirali sukobi kada je potreban istodobni pristup.

Dodatna komponenta dostupnosti smanjuje količinu zastoja potrebnu za izvršavanje administrativnih zadataka. Što DBA može brže obavljati administrativne zadatke koji zahtijevaju da baze podataka budu isključene, to će podaci biti dostupniji. Sve više dobavljači DBMS-a i neovisni dobavljači softvera (ISV-ovi) pružaju neslužbene programe koji

se mogu izvoditi na bazama podataka, dok aplikacije čitaju i pišu s njih. Ali, oni obično zahtijevaju veću vještinu i unaprijed planirano provođenje.

DBA mora razumjeti sve ove aspekte dostupnosti i osigurati da svaka aplikacija prima odgovarajuću razinu dostupnosti za svoje potrebe.

1.4 Sigurnost i autorizacija baze podataka

Nakon što je baza podataka oblikovana i implementirana, programeri i korisnici će morati pristupiti i izmijeniti podatke u bazi podataka. Ali samo ovlašteni programeri i korisnici trebaju imati pristup kako bi spriječili narušavanje sigurnosti i nepravilnu izmjenu podataka. Odgovornost administratora je da osigura da su podaci dostupni samo ovlaštenim korisnicima.

Tipično, iako ne uvijek, DBA radi s unutarnjim sigurnosnim značajkama DBMS-a u obliku SQL naredbi GRANT i REVOKE, kao i bilo kojim značajkama grupnog odobrenja DBMS-a. Mnoge radnje koje zahtijevaju okruženje baze podataka moraju biti osigurane, a to su:

- Stvaranje objekata baze podataka, uključujući baze podataka, tablice, poglede i strukturu programa.
- Promjena strukture objekata baze podataka.
- Pristup katalogu sustava.
- Čitanje i izmjena podataka u tablicama.
- Stvaranje i pristup korisnički definiranim funkcijama i tipovima podataka.
- Pokretanje pohranjenih procedura.
- Pokretanje i zaustavljanje baza podataka i pridruženih objekata baze podataka.
- Postavljanje i izmjena DBMS parametara i specifikacija.
- Pokretanje pomoćnih programa baze podataka kao što su LOAD, RECOVER i REORG.

Sigurnost baze podataka može se provoditi i na druge načine. Na primjer, pogledi se mogu stvoriti da bi blokirali pregledavanje osjetljivih stupaca ili redaka od strane krajnjih korisnika i programera. DBA također se često sučeljava s vanjskim sigurnosnim metodama koje utječe na sigurnost baze podataka.

DBA mora razumjeti i biti sposoban provoditi bilo koji aspekt sigurnosti koji utječe na pristup bazama podataka.

1.4.1 Centralizacija sigurnosti

Neke su organizacije poduzele korake za sastavljanje svih sigurnosnih i autorizacijskih zadataka, pravila i postupaka u centraliziranoj IT sigurnosnoj grupi. Takav poduhvat nije trivijalan i, kao takav, češći je u većim organizacijama nego u manjim.

Jako regulirane industrije mogu se odlučiti za centraliziranje sigurnosnih operacija. Nadalje, organizacije sa središnjim računalima imaju tendenciju za centralizaciju češće nego oni bez.

Uspješno prenošenje odgovornosti za sigurnost baze podataka iz grupe za upravljanje bazom podataka na centraliziranu sigurnosnu skupinu zahtijeva obuku sigurnosnog osoblja u tehnikama sigurnosti baze podataka. Osim toga, mnoge od tih organizacija koriste softver koji uklanja sigurnosne operacije iz DBMS-a i oponaša istu funkcionalnost u više tradicionalnom sigurnosnom paketu (kao što je RACF ili ACF2 na središnjem računalu).

Čak i ako uzmemo u obzir ova pitanja, većina organizacija se još uvijek oslanja na DBA za upravljanje sigurnošću baze podataka.

1.4.2 Upravljanje i usklađenost s propisima

Osiguravanje usklađenosti s industrijskim i državnim propisima dodatni je zadatak administracije baze podataka, barem u smislu provedbe odgovarajućih kontrola. DBA mora raditi s upravom, revizorima i poslovnim stručnjacima kako bi razumio propise koji se odnose na njihovu industriju i način na koji se podaci obrađuju.

Određeni aspekti usklađenosti s propisima odnose se na standardne operativne postupke DBA. Na primjer, propisi mogu sadržavati jezik kojim se provode posebni sigurnosni i autorizacijski postupci provođenja, zahtjeve za revizijom, specifikacije za sigurnosno kopiranje podataka i postupke upravljanja promjenama. Kako bi se osigurala sukladnost, može biti potrebna stroža dokumentacija, ili možda veći stupanj marljivosti ili automatizacije (kao što je dublje praćenje revizije).

Drugi aspekti usklađenosti s propisima mogu zahtijevati da DBA usvoji različite tehnike, taktike i vještine. Na primjer, propisi o čuvanju podataka mogu zahtijevati da se podaci čuvaju dugo nakon što ih je potrebno pohraniti u proizvodnu bazu podataka, što zahtijeva vještine arhiviranja baze podataka. Ili određene podatke možda treba zaštititi od pogleda, zahtijevajući maskiranje podataka ili, u nekim slučajevima treba postaviti šifriranje.

DBA ne bi trebao biti zadužen za razumijevanje propisa u bilo kojoj dubini, niti bi trebao postavljati standarde prema kojima se organizacija pridržava propisa. Međutim, DBA će se uključiti u pomaganje pri postavljanju odgovarajućih kontrola i postupaka za projekte usklađenosti, posebno u pogledu obrade podataka.

1.5 Sigurnosno kopiranje i oporavak

DBA mora biti spreman za oporavak podataka u slučaju problema. "Problem" može značiti sve, od greške u sustavu ili pogreške programa do prirodne katastrofe koja zatvara organizaciju. Danas se većina oporavaka javlja kao posljedica pogrešaka u aplikacijskom softveru i ljudske pogreške. Hardverski kvarovi nisu toliko rasprostranjeni kao prije. Zapravo, procjene analitičara pokazuju da je 80% grešaka u primjeni uzrokovano greškama u softveru i ljudskim pogreškama. DBA mora biti spreman vratiti podatke na upotrebljivu točku, bez obzira na uzrok i to učiniti što je brže moguće.

Prvi tip oporavka podataka koji se obično pojavljuje je oporavak na trenutnu, obično u slučaju velikog kvara. Krajnji rezultat oporavka je da je baza podataka vraćena u svoje trenutno stanje u vrijeme kvara. Aplikacije su potpuno nedostupne dok se ne dovrši oporavak.

Druga vrsta tradicionalnog oporavka je oporavak u trenutku. Oporavak u trenutku obično se provodi kako bi se riješio problem na razini aplikacije. Konvencionalne tehnike za izvođenje oporavka u trenutku će ukloniti učinke svih transakcija od određene točke u vremenu. To ponekad može uzrokovati probleme ako je tijekom tog vremenskog okvira bilo valjanih transakcija koje se moraju primijeniti.

Oporavak transakcija je treći tip oporavka koji rješava nedostatke tradicionalnih vrsta oporavka kao što su zastoji i gubitak dobrih podataka. Prema tome, oporavak transakcije je aplikacija koja obnavlja učinke određenih transakcija tijekom određenog vremenskog okvira iz baze podataka. Stoga se oporavak transakcija ponekad naziva oporavak aplikacije.

Većina stručnjaka razmišlja o oporavku kako bi riješila katastrofe kao što su hardverski kvarovi. Iako se hardverski kvarovi još uvijek događaju i tehničari moraju biti spremni da se oporave od takvih neuspjeha danas je većina oporavaka potrebna zbog ljudskih pogrešaka ili pogrešaka u programima.

DBA mora biti spreman da se nosi sa svim ovim vrstama oporavka. To uključuje razvoj strategije sigurnosnog kopiranja kako bi se osiguralo da se podaci ne izgube u slučaju pogreške u softveru, hardveru ili ručnom procesu. Strategija mora biti primjenjiva na obradu baze podataka, tako da mora sadržavati kopije slika datoteka baza podataka kao i plan za oporavak baze podataka. Potrebno je uzeti u obzir bilo koju datoteku koja nije iz baze podataka, a koja može utjecati i na aplikacije za baze podataka.

1.6 Osiguravanje integriteta podataka

Baza podataka mora biti oblikovana tako da pohrani ispravne podatke na ispravan način bez da se podaci oštete. Da bi se osigurao ovaj proces, DBA implementira pravila integriteta koristeći značajke DBMS-a. Tri aspekta integriteta su relevantna za našu raspravu o bazama podataka, a to su: fizički, semantički i unutarnji.

Fizičkim se problemima može rukovati pomoću DBMS značajki kao što su domene i tipovi podataka. DBA bira odgovarajući tip podataka za svaki stupac svake tablice. Ova radnja osigurava da su u bazu podataka pohranjeni samo podaci te vrste. To jest, DBMS provodi integritet podataka u odnosu na njihov tip. Stupac definiran kao "cijeli broj" može sadržavati samo cijele brojeve. Pokušaji pohranjivanja nenumeričkih ili neintegriranih vrijednosti u stupac definiran kao cijeli broj neće uspjeti. DBA može također koristiti ograničenja za daljnje ocrtavanje tipa podataka koji se može spremiti u stupce baze podataka. Većina relacijskih DBMS proizvoda pruža sljedeće vrste ograničenja:

- *Referencijalna ograničenja* koriste se za određivanje stupaca koji definiraju bilo koji odnos između tablica. Referencijalna ograničenja koriste se za provedbu referencijalnog integriteta, što osigurava da su sve namjerne reference iz podataka u jednom stupcu (ili skupu stupaca) tablice valjane s obzirom na podatke u drugom stupcu iste ili različite tablice.

- *Ograničenja jedinstvenosti* osiguravaju da se vrijednosti za stupac ili skup stupaca pojavljuju samo jednom u tablici.

- *Ograničenja zasnovana na provjeri* koriste se za postavljanje složenijih pravila integriteta na stupac ili skup stupaca u tablici. Ograničenja zasnovana na provjeri obično se definiraju pomoću SQL-a i mogu se koristiti za definiranje vrijednosti podataka koje su dopuštene za stupac ili skup stupaca.

Semantički integritet je teže kontrolirati i definirati. DBA-ovi moraju biti spremni provoditi politike i prakse kako bi osigurali da su podaci pohranjeni u njihovim bazama podataka točni, primjereni i upotrebljivi. Primjer semantičkog problema je kvaliteta podataka u bazi podataka. Jednostavno pohranjivanje podataka koji zadovoljavaju definicije fizičkog integriteta koji je naveden u bazi podataka nije dovoljno. Kako bi se osigurala kvaliteta podataka, potrebno je uspostaviti postupke i prakse. Primjerice, baza podataka korisnika koja sadrži pogrešnu adresu ili telefonski broj za 25% korisnika pohranjenih u njoj primjer je baze podataka loše kvalitete. Ne postoji sustavna, fizička metoda osiguranja točnosti podataka. Kvaliteta podataka se potiče putem pravilnog koda aplikacije, dobrih poslovnih praksi i specifičnih politika podataka. Redundancija je još jedno semantičko pitanje. Ako se elementi podataka pohranjuju redundantno kroz bazu podataka, DBA bi trebao dokumentirati ovu činjenicu i raditi kako bi osigurao da su redundantni podaci sinkronizirani i točni.

Posljednji aspekt integriteta je unutarnje pitanje DBMS-a. DBMS se oslanja na unutarnje strukture i kod za održavanje veza, pokazivača i identifikatora. U većini slučajeva DBMS će obaviti dobar posao održavanja tih struktura, ali DBA mora biti svjestan njihovog postojanja i kako se nositi kada DBMS ne uspije. Unutarnji DBMS integritet je bitan u sljedećim područjima:

- *Dosljednost indeksa.* Indeks zapravo nije ništa drugo nego uređena lista pokazivača na podatke u tablicama baze podataka. Ako se iz nekog razloga indeks izvuče iz sinkronizacije s podacima, indeksirani pristup ne može vratiti odgovarajuće podatke. DBA ima na raspolaganju alate za provjeru i ispravljanje ovih vrsta pogrešaka.

- *Konzistentnost pokazivača.* Ponekad veliki multimedijski objekti nisu pohranjeni u istim fizičkim datotekama kao drugi podaci. Zbog toga, DBMS zahtijeva pokazivačke strukture da bi se multimedijski podaci sinkronizirali s podacima bazne tablice. Još jednom, ovi se pokazivači mogu izgubiti ako se ne poštuju odgovarajući postupci administracije.

- *Konzistentnost sigurnosne kopije.* Neki DBMS proizvodi povremeno uzimaju neprikladne sigurnosne kopije koje se ne mogu učinkovito koristiti za oporavak. Nužno je identificirati te scenarije i poduzeti potrebne mjere.

Sveukupno gledajući, osiguravanje integriteta je bitna vještina administratora baze podataka.

Poglavlje 2

DBA alati

Svaki veliki DBMS proizvod pruža kompletan, funkcionalan sustav za upravljanje bazom podataka koji se može koristiti za pohranu i upravljanje podacima. Iako organizacije mogu instalirati i koristiti DBMS, mnogi će brzo uvidjeti da funkcionalnost potrebna za adekvatnu podršku razvoju velikih baza podataka nije osigurana samo od strane DBMS proizvoda.

Administracija i održavanje baze podataka i aplikacija je dugotrajna ako koristite samo standardne značajke DBMS-a. Srećom, mnogi DBA alati koji poboljšavaju funkcionalnost sustava za upravljanje relacijskom bazom podataka dostupni su od dobavljača trećih strana.

DBA alati smanjuju količinu vremena, truda i ljudske pogreške u održavanju učinkovitih sustava i baza podataka. Takvi alati olakšavaju administrativno opterećenje i smanjuju mogućnost pogreške. Potreba za tim alatima je vidljiva s obzirom na veliki broj proizvoda koji su dostupni. Većina organizacija implementira barem jedan dodatak za svoj DBMS. Dostupne su mnoge vrste alata koji ispunjavaju tržišne niše koje ne podržavaju glavni proizvođači DBMS-a. Ostatak ovog poglavlja daje pregled kategorija i vrsta proizvoda dostupnih DBA-u kao pomoć u upravljanju i održavanju baza podataka.

2.1 Modeliranje i oblikovanje podataka

Alati za modeliranje i oblikovanje baza podataka osiguravaju dosljedan i koherentan način stvaranja konceptualnih i logičkih modela podataka i pretvaranje istih u fizičke baze podataka. Alati za modeliranje i oblikovanje baza podataka ne moraju biti specijalizirani za određenu bazu podataka. Alati razvijeni posebno za podršku određenom DBMS-u mogu značajno smanjiti razvojno vrijeme automatiziranjem zadataka koji se ponavljaju i provjerom valjanosti modela. Međutim, ako u svojoj organizaciji upotrebljavate više DBMS proizvoda, bolje ćete odabrati alat koji može podržati sve njih, umjesto da odaberete više

alata prilagođenih određenoj bazi podataka. Prilikom odabira alata za modeliranje i oblikovanje potražite onaj koji može:

- Podržati standardne zadatke povezane s konceptualnim modeliranjem podataka kao što su dijagrami entiteta i veza i normalizacija.
- Stvoriti fizički model podataka prilagođen svakoj od korištenih DBMS platformi. Ovaj model treba podržavati sve značajke svakog DBMS-a. Na primjer, za DB2 glavno računalo trebalo bi biti u stanju prikazati sve DB2 objekte, referencijalni integritet, VCAT (volumenski katalog) i STOGROUP-definirane tablične prostore, sistemski upravljanu pohranu i planiranje kapaciteta.
- Osigurati ekspertni sustav za provjeru točnosti fizičkog modela podataka i predložiti alternativna rješenja.
- Usporediti logički model s fizičkim, uzimajući tekst koji podržava fizičke dizajnerske odluke kao što su denormalizacija i tip tablice.
- Automatski generirati standardni DDL (*Data Definition Language*) kako biste u potpunosti implementirali bazu podataka definiranu u fizičkom modelu podataka.

2.2 Upravljanje promjenama baze podataka

Unos promjena u vaše baze podataka može biti težak posao koji je sklon greškama. Međutim, rijetke su baze podataka koja ne trebajuu prolaziti kroz neke promjene tijekom korisnog vijeka trajanja. Dostupan je niz alata koji pomažu DBA-u u upravljanju i provođenju promjena u bazi podataka. Ovi alati optimiziraju i automatiziraju višestruke zadatke upravljanja promjenama, uključujući izmjenu baze podataka, usporedbu baza podataka, autorizaciju sigurnosti, praćenje revizije, upravljanje prostorom i upravljanje referencijalnim integritetom.

2.2.1 Alati za upravljanje promjenama

Prevladavajući oblik alata za upravljanje promjenama je alat za izmjenu i usporedbu baza podataka. Iako se struktura relacijskih baza podataka može modificirati korištenjem ALTER izraza, ova naredba je funkcionalno osiromašena u većini DBMS proizvoda. U teoriji, DBA bi trebao biti u stanju promijeniti sve parametre koji se mogu specificirati za objekt kada je stvoren, ali niti jedan trenutni DBMS proizvod to ne podržava. Na primjer, većina DBMS proizvoda omogućuje dodavanje stupaca postojećoj tablici, ali samo na kraju. Nadalje, oni ne dopuštaju da DBA ukloni stupce iz tablice - umjesto toga, tablica se mora brisati, a zatim ponovno stvoriti bez specificiranih stupaca.

Drugi problem s kojim se DBA-i susreću prilikom izmjene struktura baze podataka je kaskadni DROP efekt. Ako promijenite objekt baze podataka on se uklanja i ponovo stvori, ali i svi zavisni objekti se uklanjaju kada se objekt baze podataka ukloni. To uključuje tablice, sve indekse na tablicama, sve primarne i strane ključeve, sve srodne sinonime i poglede, sve okidače, sve autorizacije i, naravno, podatke. Mnoge vrste promjena objekata baze podataka ne mogu se izvesti pomoću generičkog izraza ALTER. Ovisno o DBMS-u, možda se neće moći:

- Promjeniti nazive baze podataka, tablice, pseudonima, pogleda, stupca, prostora tablice, okidača, pohranjene procedure, korisnički definirane funkcije, odnosa ili indeksa
- Premjestiti tablicu iz jedne baze podataka u drugu
- Prerasporediti redoslijed stupaca
- Promijeniti vrstu podataka i duljinu stupca
- Ukloniti stupce iz tablice
- Promijeniti primarni ključ bez brisanja i dodavanja primarnog ključa
- Dodati stupce pogledu ili uklonite stupce iz pogleda
- Promijeniti SELECT izraz na kojem se temelji pogled
- Promijeniti stupce indeksiranja
- Promijeniti specifikaciju jedinstvenosti indeksa

Takav popis obično pruža sva opravdanja potrebna za dobivanje alata za izmjenu baze podataka. Naravno, točan popis će se razlikovati od DBMS-a do DBMS-a. Alati za izmjenu baza podataka osiguravaju integrirano okruženje za izmjenu objekta baze podataka. Takvi alati obično pružaju sučelje koje se pokreće putem izbornika ili sučelje ”pokaži i klikni” koje DBA-u omogućuje da odredi vrstu potrebne promjene. Teret osiguravanja ispravne promjene baze podataka premješta se iz baze podataka u alat. U najmanju ruku, alat ALTER trebao bi:

- Zadržati ili ponovno primijeniti sve ovisne objekte, ovlaštenja i podatke na koje utječe ALTER ako je potrebno uklanjanje.
- Hijerarhijski se kretati od objekta do objekta.
- Osigurati modifikaciju utemeljenu na GUI-u koja prikazuje definicije objekata prije i poslije promjena.

- Analizirati promjene kako bi osigurao da zatražene promjene ne krše nikakva DDL pravila.
- Osigurati sposobnost praćenja promjena kako se primjenjuju.

Automatizirano rješenje za upravljanje promjenama omogućuje DBA-u da se usredotoči na potrebnu promjenu umjesto na detalje o tome kako DBMS implementira takvu promjenu. Alat je izgrađen ne samo za discipline upravljanja promjenama nego i za razumijevanje DBMS-a u kojem će se vršiti promjene. Ova ugrađena inteligencija prebacuje teret osiguravanja da promjena objekta baze podataka ne uzrokuje druge implicitne promjene od baze podataka do alata. Nadalje, nakon što je promjena identificirana i implementirana za jedan sustav, lako se može postaviti na druge kopije baze podataka s možda minimalnim promjenama. Još jedna prednost alata za upravljanje promjenama je u analizi i planiranju baza podataka. Utjecaj promjena može se ispitati prije provedbe bilo kakve promjene. To je neprocjenjiv resurs za osiguravanje sigurnih i učinkovitih promjena u bazi podataka. Ovaj tip alata također koristi automatizaciju kako bi smanjio resurse potrebne za implementaciju promjene baze podataka. Umjesto pisanja nove, složene skripte od početka za svaku promjenu baze podataka, DBA se može osloniti na alat za upravljanje promjenama da bi to postigao. Dostupnost aplikacija i baza podataka bit će poboljšana jer će proizvod provesti promjenu na najbrži mogući način.

Sve u svemu, proizvod za upravljanje promjenom baze podataka poboljšat će dostupnost i minimizirati pogreške.

2.2.2 Alati za usporedbu baza podataka

Tijekom vremena DBA će vršiti promjene u bazama podataka u cijeloj organizaciji. Moguće je da čak i uz automatizirano rješenje za upravljanje promjenom baze podataka, neke promjene će biti implementirane na nekim sustavima, ali ne i na drugima. DBA-i u velikim tvrtkama moraju pratiti desetke, čak i stotine ili tisuće poslužitelja baza podataka. Osiguravanje da se promjene učinkovito prenesu na sve te poslužitelje baza podataka može biti težak zadatak.

Alat za usporedbu baze podataka omogućuje DBA-u da usporedi jednu bazu podataka s drugom u smislu objekata i strukture baze podataka. Takvi alati će identificirati razlike i automatski generirati DDL kako bi baze podataka bile iste - iz strukturne perspektive, a ne iz perspektive sadržaja podataka. Alat za usporedbu baze podataka trebao bi DBA-u omogućiti sljedeće vrste usporedbi:

- Jedna baza podataka s drugom bazom podataka (na istom poslužitelju ili drugom poslužitelju)
- Baza podataka s datotekom DDL skripte

- Jedna DDL skripta s drugom datotekom DDL skripte

DBA-i koji upravljaju velikim brojem poslužitelja baza podataka trebali bi razmotriti korištenje takvih proizvoda. Kao što je ranije spomenuto, alati za usporedbu baza podataka uspoređuju samo strukturu baze podataka, a ne i sadržaj. Međutim, neki dobavljači nude alate koji mogu usporediti sadržaj jedne tablice sa sadržajem druge. Takvi su alati često korisni tijekom testiranja i otklanjanja pogrešaka aplikacijskih programa.

2.2.3 Alati za prebacivanje objekata baze podataka

Mnogi DBMS proizvodi ne pružaju značajku za prebacivanje objekata baze podataka s jednog poslužitelja baze podataka ili podsustava na drugi. Bez alata, to možete postići samo ručnim pohranjivanjem DDL CREATE izraza (i svih narednih ALTER izraza) u datoteku skripte, a zatim izvršavanjem skripte na drugom poslužitelju baze podataka. DBA bi tada morao prebaciti podatke iz izvornog sustava i učitati ga u ciljni sustav (ako su podaci migrirani, kao i shema baze podataka). Ručni postupci poput ovog su skloni pogreškama.

Alati za prebacivanje olakšavaju brzo prebacivanje objekata baze podataka iz jednog okruženja u drugo (npr. iz testa u produkciju). Iako sličan alatu za izmjenu tablice, alat za prebacivanje objekata ima minimalne mogućnosti promjena.

Prevacivanje se obično može specificirati na bilo kojoj razini. Na primjer, ako zatražite prebacivanje određene baze podataka, također možete migrirati sve ovisne objekte i sigurnost. Funkcionalnost je osigurana tako da se imena objekata baze podataka, autorizacijski ID-ovi i drugi objekti mogu preimenovati u skladu sa standardima instance, podsustava ili poslužitelja primatelja. Kada su parametri prebacivanja u potpunosti specificirani, alat kreira posao za implementaciju traženih objekata baze podataka u traženom okruženju.

Alat za prebacivanje može smanjiti vrijeme koje administratori baza podataka zahtijevaju za premještanje baza podataka iz okruženja u okruženje. Brži preokret rezultira bržim odgovorom na potrebe korisnika, čime se povećava učinkovitost poslovanja.

2.2.4 Alati za referencijalni integritet

Baze podataka koriste referencijalni integritet (RI) kako bi osigurala valjanost primarnog ključa za odnose s vanjskim ključevima. Međutim, RI se može teško upravljati i provoditi. RI alati eliminiraju ove poteškoće:

- Analiza podataka za povrede referencijalnog ograničenja integriteta sustava i sustava kojima upravlja korisnik

- Brže izvršavanje od DBMS objekta ili uslužnog programa za provjeru integriteta

- Omogućavanje podrške za dodatne vrste RI-a; na primjer, analizom primarnih ključeva za koje ne postoje strani ključevi i brisanjem retka primarnog ključa

Osim toga, dostupni su alati koji omogućuju pregled i ekstrahiranje podataka u referencijalnim skupovima. Takva mogućnost olakšava stvaranje razumnih testnih podataka korištenjem podskupa podataka u proizvodnim bazama podataka.

2.2.5 Alati za upit i analizu kataloga

Katalog sustava ili rječnik podataka sadrži mnoštvo informacija bitnih za rad DBMS-a. Informacije o svim objektima baze podataka, ovlaštenjima i oporavku pohranjuju se i održavaju u katalogu sustava. DBA-i se oslanjaju na te informacije kako bi obavili svoj posao. Sistemski katalog sastoji se od relacijskih tablica i može se ispitati pomoću SQL-a i/ili isporučenih spremljenih procedura. Neki dobavljači DBMS-a nude prikaze sistemskog kataloga koji su lakši za pretraživanje i praćenje. Kako god im se pristupilo, ove tablice pružaju informacijsku bazu za mnoge nadzorne i administrativne zadatke.

Kodiranje SQL-a svaki put kada DBA treba pristup informacijama u katalogu sustava može biti vrlo dugotrajan proces. Često DBA mora kombinirati informacije iz višestrukih tablica kataloga kako bi korisniku pružio činjenice relevantne za određeni zadatak. Štoviše, u većini slučajeva, dobavljači DBMS-a učinili su tablice kataloga teškim za razumijevanje i upite koristeći čudne konvencije imenovanja, denormalizirane strukture, neiskorištene stupce, loše odabire tipa podataka i malu dokumentaciju. Budući da dobavljači DBMS-a dodaju značajke novim verzijama svojih proizvoda, katalog sustava postaje sve teži za razumijevanje jer se novi podaci dodaju u već ružno oblikovan i implementiran katalog sustava. Kada se pohranjuju procedure i kada je pogled osiguran, pregledavanje tablica kataloga je lakše; međutim, u ovim konzerviranim ”upitima” ponekad nedostaju ključne informacije.

Dodatni alati upita mogu olakšati teret razvoja SQL upita za pristup tablicama kataloga sustava. Ponekad se ovi alati nazivaju alatima za vidljivost kataloga jer olakšavaju pristup informacijama pohranjenim u katalogu sustava. Osnovna značajka koja je zajednička svim alatima u katalogu je sposobnost traženja kataloških informacija pomoću GUI (ili panel-driven) sučelja bez korištenja SQL izraza.

Alati kataloga sustava koji pružaju samo tu razinu sposobnosti su u najboljem slučaju osnovni alati. Većina tih alata pruža mnogo više funkcionalnosti. Umjesto da samo omoguće pristup podacima, mnogi alati mogu izvršiti jedan ili više sljedećih zadataka:

- Stvarati sintaktički ispravne DDL izraze za sve objekte baze podataka čitanjem odgovarajućih tablica sistemskog kataloga. Te se naredbe općenito izvršavaju odmah ili spremaju u skup podataka za buduću referencu ili uporabu.

- Izmjeniti stupce koji se mogu ažurirati pomoću sučelja koje nije SQL.

- Izraditi sintaktički ispravne naredbe za autorizaciju / sigurnost iz kataloga na isti način na koji se generira DDL.
- Izvršiti „analize pada“ na SQL DROP izrazu. Ova analiza određuje učinak DROP-a s detaljnim opisom svih ovisnih objekata i sigurnosti koji će biti izbrisani kao rezultat izvršavanja DROP-a.
- Osigurati hijerarhijski popis objekata baze podataka.
- Raditi izravno na katalogu sustava ili na kopiji kataloga sustava kako bi smanjili prepirke u sustavu.
- Generirati cijeli skup naredbi baze podataka koje se koriste za upravljanje bazom podataka. Primjeri mogu uključivati pokretanje ili zaustavljanje baze podataka ili tabličnog prostora, prikazivanje informacija o sustavu i mnoge druge naredbe.

Ove značajke pomažu DBA-u u obavljanju svakodnevnih dužnosti. Nadalje, alat za upite u katalogu može uvelike smanjiti količinu vremena potrebnog novom DBA-u (ili novom članu osoblja) da postane produktivan član DBA tima.

2.3 Uređivači tablice

Postoje samo dvije metode ažuriranja relacijskih podataka koje isporučuje većina DBMS proizvoda a to su:

- SQL naredbe DELETE, INSERT i UPDATE
- Uslužni programi za baze podataka kao što su LOAD ili IMPORT

SQL izrazi djeluju na skup podataka odjednom, tako da jedan SQL izraz može utjecati na više redova - ili čak na sve retke. Kodiranje SQL izraza za svaku izmjenu podataka koja je potrebna tijekom faze razvoja i testiranja aplikacije može biti dugotrajno. Nadalje, uslužni programi za baze podataka kao što su LOAD i IMPORT nisu održivo sredstvo za stvaranje malih ciljanih promjena podataka. Oni su oblikovani i optimizirani za prijenos podataka naveliko.

Alat za uređivanje tablica može smanjiti vrijeme potrebno za jednostavnu izmjenu podataka pružajući mogućnost uređivanja cijelog zaslona za tablice baze podataka. Korisnik specificira tablicu za uređivanje i pokreće se uređivač tablice. Podaci se korisniku prikazuju kao niz redaka, a stupci se razdvajaju razmacima. Linija zaglavlja označava nazive stupaca. Podaci se mogu pomicati gore-dolje, kao i lijevo i desno. Za promjenu podataka korisnik jednostavno upisuje podatke preko trenutnih podataka.

Alat za uređivanje tablice može smanjiti vrijeme potrebno za jednostavnu izmjenu podataka. Ova vrsta alata idealna je za potporu procesu razvoja aplikacija. Programer može napraviti brze promjene bez kodiranja SQL-a. Osim toga, ako se ispravno implementira, uređivač tablica može smanjiti broj pogrešnih modifikacija podataka koje su napravili početnici SQL-a.

Treba biti oprezan prije korištenja uređivača tablice na kritičnim proizvodnim podacima. Kada se koristi uređivač tablice, svi stupci su dostupni za ažuriranje, a jednostavna pogreška može prouzročiti neželjena ažuriranja. Izvorni kod u SQL-u treba koristiti ako morate osigurati ažuriranje samo određenih stupaca.

2.4 Upravljanje performansom

Osiguravanje optimalne performanse jedan je od najvećih problema s kojim se DBA-i stalno suočavaju. Najglasnije pritužbe dolaze od korisnika koji moraju čekati dulje nego što su navikli čekati da njihovi programi odgovore.

To je osobito istinito ako korisnici nikada nisu morali čekati u prošlosti. Alati za upravljanje performansom pomažu DBA-u izmjeriti brzinu i učinkovitost SQL upita, struktura baza podataka i parametara sustava. Takvi alati rade u pozadini kako bi uhvatili statistiku performansi baze podataka i upozorili DBA kada dođe do problema. Napredni alati za performansu mogu poduzeti proaktivne mjere za rješavanje problema dok se događaju.

Svaka aplikacija baze podataka, u svojoj osnovi, zahtijeva tri komponente kako bi radila: sustav, bazu podataka i aplikaciju. Da bi pružio dobre performanse, DBA mora moći pratiti i podešavati svaku od tih komponenti. DBA ima dostupne alate za praćenje i optimizaciju svake od tih komponenti.

2.4.1 Alati za performansu sustava

Alati za performansu sustava pregledavaju poslužitelja baze podataka, njegovu konfiguraciju i upotrebu. Najčešće korišteni alat za performansu sustava je *monitor performansi*. Alati za praćenje i analizu performansi baze podataka podržavaju mnoge vrste zahtjeva usmjerenih na performansu. Na primjer, alati za performansu sustav mogu raditi:

- U pozadinskom načinu rada kao posao koji izvještava o statistikama performansi koje je napisao DBMS-ov alat za praćenje.

- U prvom planu kao mrežni monitor koji ili hvata informacije o tragovima ili bilježi informacije iz kontrolnih blokova DBMS-a kada se aplikacije izvršavaju.

- Uzorkovanjem jezgre baze podataka i korisničkih adresnih prostora kako se program izvodi i hvatanjem informacija o performansi posla, neovisno o tragovima baze podataka.
- Snimanjem informacije praćenja baze podataka i njihovo održavanje u datoteci povijesti za izradu povijesnih izvješća o performansi i za predviđanje trendova performanse.
- Kao uređaj za planiranje kapaciteta koji daje statističke podatke o aplikaciji i okolini u kojoj će raditi.
- Kao alat za analizu činjenica na radnoj stanici koji analizira i grafički prikazuje sve aspekte performansi aplikacija i performansi na razini cijelog sustava.

Svaki monitor performansi baze podataka podržava jednu ili više tih značajki. Vrednovanje monitora performansi baze podataka je složen zadatak.

Moderni alati za performansu baze podataka mogu postaviti pragove performansi koji će, nakon što se dostignu, upozoriti DBA, obaviti neki drugi zadatak za obavještavanje ili zapravo riješiti problem. Ovi alati su tipično na bazi agenta. *Agent* je dio neovisnog koda koji se izvodi na poslužitelju baze podataka u potrazi za problemima. On komunicira s konzolom koja se pokreće na drugom računalu koje pregledava DBA, ali se ne oslanja na nju. Ova arhitektura agenta omogućuje učinkovito praćenje baze podataka jer agent nije vezan za radnu stanicu i može djelovati neovisno. Agent šalje informacije DBA-u samo kada je to potrebno.

Osim toga, dostupni su neki alati za performanse sustava koji se fokusiraju na određenu komponentu DBMS-a, kao što je podatkovna predmemorija. Takav se alat može koristiti za modeliranje memorijskih zahtjeva, za predmemoriranje baze podataka, za hvatanje statistike upotrebe predmemorije podataka, a možda čak i za izradu preporuka za poboljšanje učinkovitosti podatkovne predmemorije.

Drugi tip alata za optimizaciju performansi omogućuje promjenu konfiguracijskih parametara baze podataka bez ponovnog pokretanja instance, podsustava ili poslužitelja DBMS-a. Ovi alati su korisni kada se zahtijeva da se DBMS zaustavi i ponovno pokrene. Takvi alati mogu dramatično poboljšati dostupnost, posebno ako konfiguracijske parametre treba često mijenjati, te ako DBMS ne podržava dinamičko modificiranje parametara.

2.4.2 Alati za performansu baze podataka

Većina DBMS-ova ne pruža mogućnost analize inteligentne baze podataka. Umjesto toga, DBA ili analitičari performansi moraju koristiti poglede i upite kataloga sustava, ili sustav kataloga sustava, kako bi pratili svaku bazu podataka i njezine objekte. To nije optimalno rješenje jer se oslanja na ljudsku intervenciju za učinkovitu organizaciju baze podataka,

otvarajući mogućnost ljudske pogreške. Srećom, dostupni su alati za analizu baze podataka koji mogu proaktivno i automatski pratiti vaše okruženje baze podataka. Ovi alati za analizu baza podataka obično mogu:

- Prikupljati statističke podatake za tablice i indekse: standardne statističke informacije, proširene statistike koje obuhvaćaju više informacija (na primjer, opseg skupova podataka) ili kombinaciju oboje
- Pročitati temeljne skupove podataka za objekte baze podataka za snimanje trenutnih statistika, čitati statistike baze podataka iz sistemskog kataloga, čitati jedinstvene tablice za alat koji je obuhvatio poboljšanu statistiku, ili bilo koju njihovu kombinaciju
- Postavljati pragova na temelju statistike baze podataka pri čemu se može pozvati automatsko planiranje reorganizacije baze podataka i drugih zadataka održavanja

2.4.3 Alati za performansu aplikacija

Pisanje SQL izraza za pristup tablicama baze podataka odgovornost je tima za razvoj aplikacija. Međutim, DBA se obično uključuje kada je riječ o performansi SQL-a. Uz fleksibilnost SQL-a, isti se zahtjev može izvršiti na različite načine. Budući da su mnoge od tih metoda neučinkovite, performanse aplikacija mogu divlje varirati, osim ako SQL ne bude analiziran i podešen od strane stručnjaka prije implementacije. Čak 80% svih problema s performansama baze podataka uzrokovano je neučinkovitim SQL i aplikacijskim kodom.

Naredbe EXPLAIN ili SHOWPLAN daju informacije o pristupnim stazama koje koriste SQL upiti raščlanjivanjem SQL-a u aplikacijskim programima i stavljanjem kodiranog izlaza u PLAN_TABLE ili izradom standardnog izvješća o pristupnoj stazi. Za mjerenje performanse, DBA mora dekodirati te podatke i odrediti je li dostupan učinkovitiji pristupni put.

prolazak kroz aplikaciju trebao bi pregledati sve SQL izraze, odabrane pristupne putove i programski kod u koji je ugrađen SQL.

Pregledi SQL koda potrebni su kako bi se osiguralo korištenje optimalnih tehnika dizajniranja SQL-a. Prolaskom kroz aplikaciju trebale bi se pregledati sve SQL naredbe, odabrani pristupni putevi i programski kod u koji je ugrađen SQL. Pregled također uključuje procjenu statističkih podataka u bazi podataka kako bi se utvrdilo jesu li korištene statistike na razini proizvodnje u vrijeme EXPLAIN-a. Pregled izvornog koda aplikacije liniju po liniju je zamoran i sklon pogreškama, a može uzrokovati zaostatke aplikacija. SQL alati za analizu uvelike pojednostavljuju ovaj proces. SQL alat za analizu obično:

- Analizira SQL u aplikacijskom programu, opisujući pristupne staze odabrane u grafičkom formatu, na engleskom ili oboje.

- Izdaje upozorenja kada naiđu specifični SQL konstrukti. Na primjer, svaki put kada se traži sortiranje (prema ORDER BY, GROUP BY ili DISTINCT), poruka obavještava korisnika o potrebnoj vrsti.
- Predlaže alternativna SQL rješenja koja se temelje na "ekspertnom sustavu" koji čita SQL izraze i njihove odgovarajuće PLAN_TABLE stavke i predstavlja alternativne SQL opcije.
- Proširuje pravila koja koristi "ekspertni sustav" za snimanje pravila specifičnih za određeno mjesto.
- Analizira na razini podsustava, instance, poslužitelja, aplikacije, plana, paketa ili SQL izraza.
- Pohranjuje više verzija programa EXPLAIN, stvara usporedbe performanse i planira izvješća o povijesti.

SQL alati za analizu mogu automatizirati glavne dijelove procesa pregledom koda. Dostupni su i alati koji analiziraju performansu aplikacijskog koda u kojem je SQL ugrađen. Ovi alati obično obuhvaćaju detaljne informacije o programima dok se izvode i pružaju izvješća koja određuju koja područja koda troše najviše resursa.

2.5 Sigurnosno kopiranje i oporavak

Osiguravanje oporavka sustava baza podataka je složen zadatak. Da bi se posao obavio na odgovarajući način, DBA-i trebaju razumjeti značajke sigurnosnog kopiranja i oporavka DBMS-a, kako se povezuju s diskovnim sustavima za pohranu podataka, te poslovni učinak podataka na organizaciju. Ponekad DBA-i trebaju pomoć.

Srećom, postoje brojni alati koji pojednostavljuju i automatiziraju proces sigurnosnog kopiranja i oporavka. Najjednostavniji oblik alata za sigurnosno kopiranje i oporavak je uslužni program velike brzine. ISV alati koji ubrzavaju proces izrade sigurnosnih kopija slika i korištenja tih sigurnosnih kopija za oporavak mogu se koristiti za smanjenje zastoja i povećanje dostupnosti podataka. Neki alati pojednostavljuju, a ne samo ubrzavaju te procese. Na primjer, mnogi uslužni programi za sigurnosno kopiranje i oporavak velike brzine mogu izraditi sigurnosne kopije cijelih baza podataka ili objekata baze podataka tehnikama maskiranja i zamjenskih znakova. Na primjer, razmotrite sljedeću naredbu: COPY GL21DBX2.T *. Naredba kao što je ova može se koristiti za izradu sigurnosne kopije kopija svake tablice u bazi podataka pod nazivom GL21DBX2 koja počinje slovom T.

Dostupni su dodatni proizvodi koji automatiziraju cijeli proces oporavka. Takvi alati mogu prihvatiti zahtjev za oporavkom, pregledati zapisnik i kopirati kopije slika, napraviti

preporuku o tome kako se oporaviti, mogu čak i izgraditi skripte za izvođenje oporavka koje bi rezultirale s većinom prikupljenih podataka u najmanjoj količini zastoja.

Konačna vrsta alata za oporavak je alat za oporavak temeljen na zapisima. Takav se alat može koristiti za pregled dnevnika i izradu povratnog SQL-a. Na primjer, pretpostavimo da je DBA pogreškom izdao DELETE izraz. Pomoću alata za oporavak na temelju dnevnika, DBA bi unio ime programa i vrijeme u kojem je bio pokrenut. Alat bi pregledao dnevnik, pronašao DELETE i kreirao INSERT izraze da bi reproducirao izbrisane podatke. Jednostavnim pokretanjem naredbe INSERT-a, baza podataka se oporavlja. Naravno, ovo je pojednostavljen primjer, ali alat za analizu temeljen na zapisima može poslužiti kada je potrebno brzo identificirati i ispraviti pogrešne izmjene baze podataka.

2.6 Uslužni programi za baze podataka

Mnogi uslužni programi za baze podataka koji se slobodno isporučuju s DBMS-om poznati su po lošoj performansi. Međutim, ovi uslužni programi moraju popuniti, administrirati i organizirati vaše baze podataka. Tipični uslužni programi koji se pružaju su LOAD, UNLOAD, REORG, BACKUP i RECOVER, kao i uslužni programi za provjeru integriteta.

Proizvođači treće strane osiguravaju alate za podršku koji zamjenjuju uslužne programe za baze podataka i pružaju istu ili više funkcionalnosti na učinkovitiji način. Na primjer, nije nečuveno da dobavljači trećih strana tvrde da njihovi uslužni programi izvršavaju radnje od četiri do deset puta brže od izvornih DBMS alata. Te tvrdnje moraju biti potkrijepljene podacima i aplikacijama u vašoj organizaciji. Prije nego što se obaveže bilo kojem uslužnom programu treće strane, DBA mora biti siguran da proizvod udovoljava najmanje sljedećim zahtjevima:

- Ne ruši integritet podataka u bazi podataka.
- Pruža minimalno iste značajke kao odgovarajući izvorni uslužni program. Na primjer, ako uslužni program REORG reorganizira indekse i prostore tablica, poboljšani REORG alat mora biti u mogućnosti to učiniti.
- Pruža vrijeme izvršenja najmanje dvostruko brže od odgovarajućeg uslužnog programa baze podataka. Na primjer, ako uslužni program Sybase DUMP zahtijeva 20 minuta za izradu sigurnosne kopije tablice, poboljšani alat za izradu sigurnosne kopije mora odbaciti istu tablicu za najviše 10 minuta. Naravno, to ne mora biti čvrsto pravilo. Ponekad je čak i umjereno povećanje vremena obrade dovoljno da se opravda trošak ponude treće strane.

- Ispravlja nedostatke u standardnim uslužnim programima za baze podataka. Na primjer, ako uslužni program LOAD ne učitava podatke slijedom grupiranja indeksa, zamjenski uslužni program LOAD mora to učiniti.

Prilikom testiranja korisnih alata različitih proizvođača, obavezno se trebaju provesti testovi. Na primjer, uvijek ponovno učitajte ili oporavite prije testiranja REORG alata, inače možete iskriviti svoje rezultate zbog različitih razina organizacije tablice. Osim toga, uvijek pokrenite testove za svaki alat na istom objektu s istom količinom podataka i provjerite je li podatkovna predmemorija prazna između svakog testiranja. Konačno, provjerite je li radno opterećenje na sustavu jednako (ili što je moguće bliže) prilikom testiranja svakog proizvoda jer istodobno opterećenje može iskriviti rezultate testova.

2.6.1 Uslužni alati za upravljanje

Druga vrsta baze podataka koja nudi uslugu je upravitelj usluga. Ovaj tip alata pruža administrativnu podršku za stvaranje i izvršavanje poslova uslužnog programa baze podataka. Ovi alati za generiranje i upravljanje alatima:

- Automatski generiraju parametre uslužnog programa, JCL ili naredbene skripte.
- Nadziru uslužne programe baza podataka koji se izvršavaju.
- Automatski raspoređuju uslužne programe kada se aktiviraju iznimke.
- Ponovno pokreću uslužne programe s minimalnom intervencijom.

2.7 Alati za zaštitu podataka, upravljanje, rizik i usklađenost

Sigurnost i zaštita baza podataka, zajedno s zahtjevima upravljanja, rizika i usklađenosti (GRC) nameću potrebu za upravljanjem sigurnošću i usklađenošću. Postoje alati koji se mogu koristiti za poboljšanje sigurnosti i usklađenosti baze podataka, od kojih su neki važni DBA-u za razumijevanje i korištenje. S druge strane, GRC je poslovni prijedlog, a većina rješenja u ovoj kategoriji nije relevantna za DBA-e. DBA mora biti svjestan alata koji se integriraju s DBMS-om i njihovim bazama podataka ili nametnuti dodatne zahtjeve u smislu administrativnog radnog opterećenja ili učinaka.

2.7.1 Alati za reviziju

Revizija je ispitivanje prakse kojom se utvrđuje njezina ispravnost. Softver za reviziju baze podataka pomaže u praćenju kontrole podataka, definiranja podataka i integriteta podataka

u okruženju baze podataka. Većina DBMS proizvoda pruža ograničene mehanizme revizije, ali te su značajke obično teške za korištenje i održavanje. Alati za reviziju pružaju mogućnost revizije na granularnoj razini.

Tipična oprema za reviziju dopušta reviziju na različitim razinama unutar DBMS-a, na primjer, na razini baze podataka, na razini objekta baze podataka i korisničkim razinama. No, hvatanje toliko informacija, osobito u opterećenom sustavu, može uzrokovati lošiju performansu. Izrada potrebnih detalja revizije mora se postići bez umanjenja rada kompjutoriziranih sustava koji održavaju funkcioniranje prakse.

Detalji i vjerodostojnost proizvedenog revizijskog zapisa jednako su važni kao i performanse operativnih sustava. Kontrolni tragovi moraju biti dovoljno detaljni da obuhvate ”prije” i ”poslije” slike promjena baze podataka. Ako mehanizam koji obuhvaća detalje revizije nije sveobuhvatan i učinkovito osmišljen, prestaje biti rješenje za usklađenost. Nadalje, revizijski tragovi moraju biti pohranjeni negdje, čime se štiti autentičnost revidiranih informacija, omogućujući neometan pristup izvještavanju. Zbog potencijalnog opsega promjena u podacima baze podataka, korisna mogućnost revizije mora omogućiti selektivno stvaranje revizijskih zapisa kako bi se minimizirali problemi izvedbe i pohrane.

Postoji nekoliko popularnih tehnika koje se mogu primijeniti za reviziju podataka baze podataka. Najbolja tehnika uključuje proaktivno praćenje operacija baze podataka izravno na poslužitelju baze podataka. Ova tehnika obuhvaća sve zahtjeve za podacima kako su napravljeni. Snimanjem pojedinosti revizije na razini poslužitelja, softver može jamčiti da se prati sav pristup. Druge tehnike, kao što su revizija na temelju praćenja ili parsiranje baza podataka baze podataka, mogu propustiti određene vrste aktivnosti baze podataka.

Robusno rješenje za reviziju pristupa bazi podataka koje se odnosi na usklađenost s propisima trebalo bi biti u mogućnosti dati odgovore barem na sljedeće pitanja:

- Tko je pristupao podacima?
- Na koji datum i vrijeme je bio pristup?
- Koji je programski ili klijentski softver korišten za pristup podacima?
- Na kojoj lokaciji je izdan zahtjev?
- Koji SQL je izdan za pristup podacima?
- Je li zahtjev bio uspješan i, ako je tako, koliko je redova podataka dohvaćeno?
- Ako je zahtjev izmjena, koji su podaci promijenjeni? (Slika ”prije” i ”nakon” promjene bi trebala biti dostupna.)

Naravno, iza svakog od tih pitanja stoji mnogo detalja. Robusno rješenje za reviziju pristupa bazi podataka trebalo bi osigurati neovisan mehanizam za dugoročno pohranjivanje i pristup pojedinostima revizije. Rješenje treba ponuditi upite za najčešće tipove upita,

ali informacije o reviziji trebale bi biti dostupne pomoću standardnih alata za upite kako bi revizorima bilo lakše prilagoditi upite prema potrebi.

Alat za reviziju trebao bi izraditi skup unaprijed pripremljenih izvješća prilagođenih industriji i vladinim propisima. Organizacije sa strogim sigurnosnim i revizijskim zahtjevima, ili sa značajnim regulatornim ili industrijskim zahtjevima, trebaju razmotriti korištenje alata za reviziju baze podataka zbog slabih revizijskih mogućnosti većine DBMS proizvoda.

2.7.2 Alati za praćenje

Alati za praćenje obično se oslanjaju na tehnologiju revizije baze podataka kako bi gledali pristup bazama podataka i uspostavili trendove i obrasce. Takvi su alati korisni za identifikaciju i zabranu ponašanja izvan norme za proaktivno zaustavljanje napada na baze podataka.

Na primjer, razmislite o korisniku koji svakodnevno radi na osjetljivim podacima. Alat za praćenje može utvrditi normalne obrasce korisnika. Možda osoba općenito pristupa osjetljivim podacima između 9:00 i 11:00 ujutro i opet između 14:00 i 16:00 poslijepodne. Korisnik to čini dosljedno, manje ili više, svakog radnog dana. Zatim, u subotu navečer, isti korisnik pokušava pristupiti osjetljivim podacima za sve korisnike u sjeveroistočnoj regiji. To zvuči kao potencijalni sigurnosni problem. Alat za praćenje trendova može snimiti i izvijestiti o tome tako da se atipični zahtjev dokumentira. Ili, za organizacije koje su opreznije, alat za praćenje može biti u mogućnosti zabraniti takav nestandardni pristup, zahtijevajući autorizacijsko odobrenje prije nego što mu dopusti nastavak.

2.7.3 Alati za profiliranje podataka

Još jedna korisna kategorija alata za upravljanje podacima je mogućnost profiliranja podataka. Profiliranje vaših podataka je metodologija za dobivanje uvida u vaše poslovne podatke i poboljšanje procesa za poboljšanje kvalitete podataka. Alat za profiliranje podataka koristi se za otkrivanje struktura podataka i anomalija u vašim podacima. Tehnike profiliranja pomažu vam da otkrijete slučajeve neprikladnih podataka gdje se podaci ne podudaraju s definicijom metapodataka, obrasci se ne podudaraju, vrijednosti su netočne, a postoje i redundantni podaci.

Na primjer, razmislite o tablici baze podataka s stupcem koji sadrži brojeve socijalnog osiguranja. Znamo da bi svi trebali biti u obliku nnn-nn-nnnn. Alat za profiliranje podataka može se primijeniti kako bi se ispitao određeni stupac kako bi se pokazao postotak podataka koji ne odgovaraju tom uzorku. Ili razmislite o drugom stupcu u kojem ne znate određeni uzorak podataka. U ovom slučaju, alat za profiliranje podataka može otkriti uzorak za koji

se smatralo da ne postoji. U svakom slučaju, alati za profiliranje podataka korisni su za čišćenje podataka.

2.7.4 Alati za maskiranje podataka

Maskiranje podataka je dodatna kategorija rješenja za zaštitu podataka koja se može koristiti za zamagljivanje stvarnih vrijednosti u bazi podataka. Možda ćete morati maskirati vrijednosti podataka za izvještavanje ili druge razloge za skrivanje osjetljivih podataka. Na primjer, vrijednosti na platnoj kartici ne mogu se pojaviti otkrivene na potvrdama, one će bit maskirane.

2.7.5 Sigurnosni alati

Sigurnost baze podataka obično se potiže unutar baze podataka korištenjem GRANT i REVOKE SQL naredbi, koji odobravaju autorizaciju eksplicitno i implicitno korisnicima baze podataka. Neki DBMS proizvodi osiguravaju izlaze za autorizaciju kako bi se omogućila komunikacija s vanjskim paketima za upravljanje sigurnošću. Time se olakšava administrativno opterećenje sigurnosti baze podataka omogućujući funkciji sigurnosti korporativnih podataka da upravlja skupinama korisnika.

Implementacija sigurnosti u većini relacijskih baza podataka ima nekoliko problema. Najvažniji među tim nedostacima je učinak kaskadnog REVOKE. Ako je ovlaštenje opozvano od jednog korisnika koji je prethodno odobrio ovlaštenje drugim korisnicima, sva ovisna ovlaštenja također su opozvana. Alat za dodatak sigurnosti baze podataka može riješiti taj problem. Ovi alati obično analiziraju učinke REVOKE. Ovi alati omogućuju korisniku opoziv ovlaštenja i preraspodjelu ovlaštenja, bilo ponovnim kreiranjem odgovarajućih GRANT iskaza kako bi ponovno primijenili ovlaštenja koja su implicitno opozvana ili ukidanjem ovlaštenja i automatskom ponovnom primjenom svih implicitnih REVOKE izraza u pozadini.

Alati za sigurnost baze podataka pružaju i druge funkcije. Razmislite o administrativnim opterećenjima kada se korisnici baze podataka zaposle, otpuste ili premjeste. Sigurnost se mora dodati ili ukloniti. Dobar sigurnosni alat omogućuje korisniku da izdaje naredbu GRANT LIKE koja može kopirati ovlaštenje baze podataka iz jednog objekta baze podataka u drugi ili iz jednog korisnika baze podataka u drugi. Pretpostavimo da je DBA prebačen u drugi odjel. Sigurnosni alat može dodijeliti sva ovlaštenja tog DBA-a drugom korisniku prije nego što opozove svoje ovlaštenje. Ili pretpostavimo da je stvorena nova tablica za postojeću aplikaciju i ona zahtijeva iste autorizacije kao i druge tablice u aplikaciji. Ovaj tip alata omogućuje korisniku da kopira svu sigurnost iz jedne tablice u novu tablicu.

Postoji još jedan tip proizvoda sigurnosti baze podataka. Umjesto povećanja sigurnosti baze podataka, *proizvod za zamjenu sigurnosti* zamjenjuje sigurnost baze podataka vanjskim paketom. Primarna prednost ovog proizvoda je ta što objedinjuje administraciju sigurnosti podataka i baze podataka. Druga prednost je da se kaskadni REVOKE efekt može eliminirati jer većina vanjskih paketa za sigurnost podataka ne kaskadira sigurnosne opozive.

Alati za zamjenu sigurnosnih baza podataka također imaju svoje slabosti. Ovi alati nisu u skladu s rigoroznom definicijom relacijskog modela, koji navodi da DBMS mora kontrolirati sigurnost. Neki ne pružaju sve vrste sigurnosti baze podataka. Na primjer, autorizacije na razini sustava često se izostavljaju. Drugi nedostatak je da ako vanjski sigurnosni paket ne uspije, podaci su nezaštićeni.

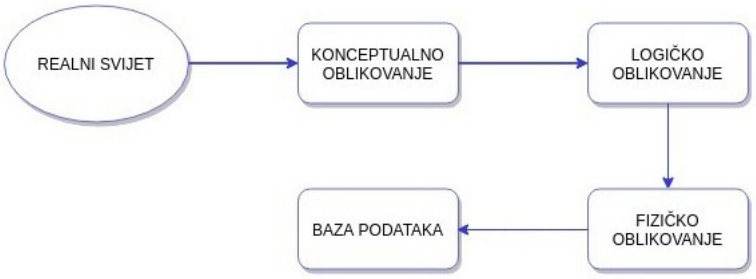
Poglavlje 3

Studijski primjer

Kako bi se kreirala neka baza podataka koja će biti podrška pri radu nekog sustava, potreban je timski rad, niz stručnjaka te primjena alata i metoda potrebnih za samo kreiranje baze. Projekt izrade baze podataka se može podijeliti u nekoliko aktivnosti, a to su: utvrđivanje i analiza zahtjeva, projektiranje, implementacija, testiranje i na kraju održavanje. Kako bi se utvrdili zahtjevi nekog sustava potrebno je proučiti sve informacijske tokove i materijale te uočiti veze među njima. Također se provodi razgovor sa korisnicima kako bi se pobliže sve moglo shvatiti. Nakon što se utvrde zahtjevi za bazu, slijedi oblikovanje baze.

Cilj oblikovanja je kreirati bazu u skladu sa zahtjevima koji su definirani u prethodnom koraku. Oblikovanje baze podataka predlaže način na koji se podaci grupiraju, povezuju i strukturiraju. Samo oblikovanje je dosta složen proces te zahtjeva neke standardne faze kreiranja baze, a to su: konceptualno, logičko te fizičko oblikovanje.

Implementacija je sljedeći korak pri izradi baze podataka, a sastoji se od fizičke realizacije oblikovane baze podataka. Pri korištenju sustava za upravljanje podataka pokreću se naredbe, ovisno o jeziku koji se koristi u nekom sustavu za upravljanje bazama podataka. Nakon što su tablice kreirane i povezane, slijedi popunjavanje tablica podacima. Na slici 3.1. su prikazana faze oblikovanja baze podataka.



Slika 3.1: Oblikovanje baze podataka

Za izradu baze podataka i svih njezinih objekata koristiti ću Microsoft SQL Server 2014. Microsoftova relacijska baza podataka (RDBMS) namijenjena je korporativnim okruženjima. Podržava Transact-SQL, ekstenziju SQL-a. Baza je dostupna u velikom broju izdanja od kojih je Express izdanje koje će se korititi u ovom radu potpuno besplatno. U SQL serveru postoje četiri tipa SQL upita: DML (Data Manipulation Language), DDL (Data Definition Language), DCL (Data Control Language) i TCL (Transaction Control Language). Svakom od ovih tipova upita pripada različit set SQL naredbi.

DML upiti koriste se za dohvat, umetanje, izmjenu te brisanje podataka korištenjem SQL naredbi SELECT, INSERT, UPDATE i DELETE.

DDL upitima kreira se i mijenja struktura objekata (tablica, pogleda, funkcija itd.) u bazi podataka, za što je potrebno koristiti SQL naredbe CREATE, ALTER i DROP.

DCL upitima definiraju se uloge i dozvole u bazi podataka što ih čini ključnim upitima za kontrolu pristupa. U tu svrhu koriste se SQL naredbe GRANT i REVOKE.

TCL upiti koriste se za upravljanje transakcijama. Njima je moguće uspješno potvrditi kraj transakcije (COMMIT) ili napraviti njen opoziv (ROLLBACK).

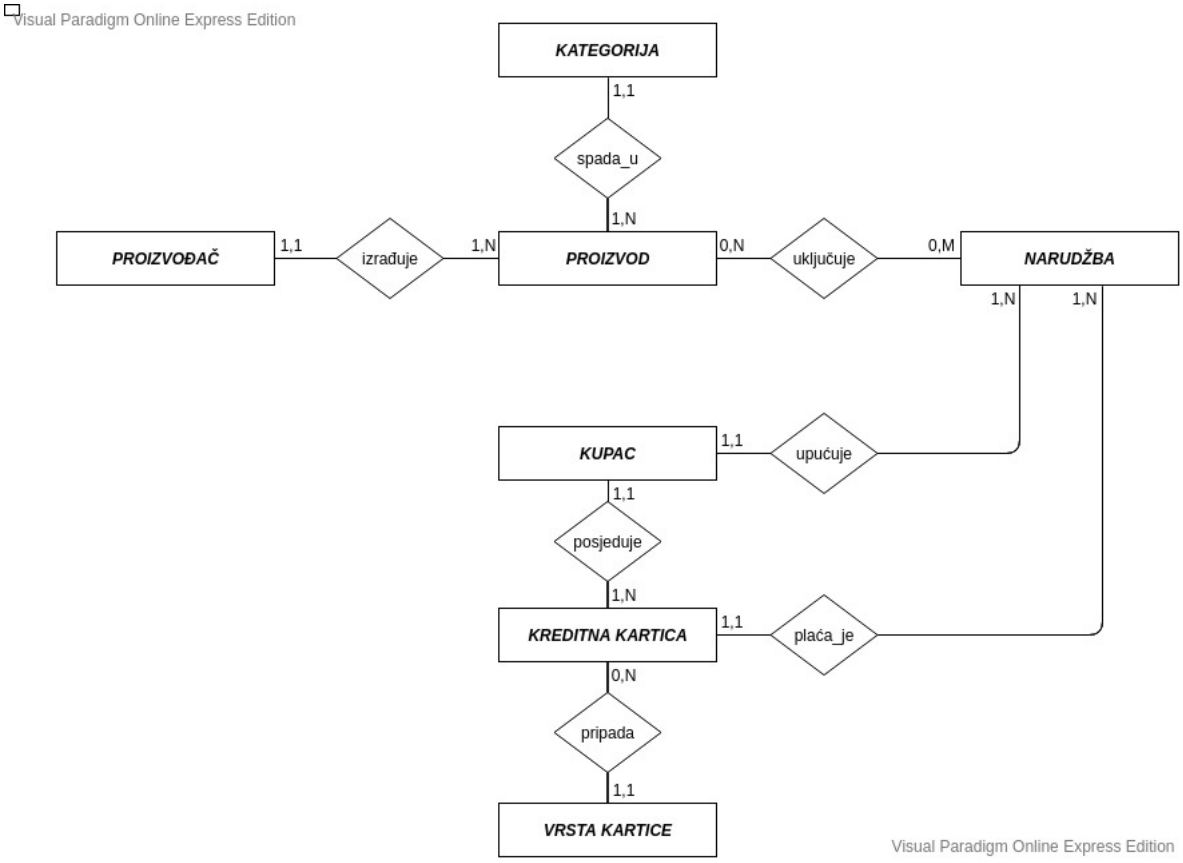
3.1 Konceptualno oblikovanje baze podataka

Prvi korak u oblikovanju konceptualne sheme je otkrivanje samih elemenata od kojih se shema sastoji. U pravilu, elementi se trebaju prepoznati čitanjem specifikacije. Imenice upućuju na atribute i entitete, a glagoli upućuju na veze. Kod prepoznavanja elemenata sheme, projektant često ne zna treba li imenicu shvatiti kao entitet ili kao atribut. Nakon što smo otkrili entitete, veze i atribute, potrebno je za svaki entitet utvrditi koji ga atributi opisuju. Moguće je da neki atributi pripadaju vezi između entiteta, a ne pojedinim entitetima. Također za svaku vezu treba odrediti njezinu funkcionalnost, obaveznosti članstva te kardinalnost. Za svaki entitet potrebno je odabrati primarni ključ. Specifikacija mora biti potpuno jasna kako bi se mogli otkriti entiteti, atributi i veze.

U nastavku rada će biti prikazana baza podataka online trgovine u kojoj je potrebno voditi evidenciju o proizvodima koji se nude, a koji su podijeljeni u različite kategorije. Za svaki proizvod će se bilježiti njegov naziv, proizvođač, broj komada kojima se raspolaže, cijena po komadu te podatak o tome je li proizvod raspoloživ ili ne. Nadalje, zahtijeva se vođenje evidencije o kupcima proizvoda i narudžbama koje oni upute. Jedna narudžba može uključivati jedan ili više proizvoda, a za svaku narudžbu treba zabilježiti datum i vrijeme njezina primitka, datum njezine obrade te podatke o tome je li narudžba izvršena. Osim toga, treba pohraniti podatke o broju jedinica proizvoda koji se naručuje, cijeni koju za njih treba platiti, te datum i vrijeme njihove isporuke. Kupac naručene proizvode plaća putem kreditne kartice, pri čemu se uz jednog kupca može vezivati više vrsta kreditnih kartica pa stoga uz narudžbu treba također zabilježiti i kojom je kreditnom karticom ona plaćena.

Iz ovih specifikacija određeni su entiteti, atributi i veze između njih, te je napravljen reducirani Chenov dijagram koji je prikazan na slici 3.2. Lista entiteta i atributa:

- Tip entiteta PROIZVOĐAČ, s atributima: id_proizvođača, naziv_tvrtke, adresa_tvrtke, kontakt, email.
- Tip entiteta PROIZVOD, s atributima: id_proizvoda, naziv, broj_jedinica, cijena_po_komadu, raspoloživ_dn.
- Tip entiteta KATEGORIJA, s atributima: id_kategorije i naziv_kategorije.
- Tip entiteta KUPAC, s atributima: id_kupca, ime, prezime, adresa, kontakt, email.
- Tip entiteta NARUDŽBA, s atributima: id_narudžbe, datum_primitka, vrijeme_primitka, datum_obrade, izvršena_dn.
- Tip entiteta KREDITNA KARTICA, s atributima: id_kartice, datum_isteka, broj_kartice.
- Tip entiteta VRSTA KARTICE, s atributima: id_vrste i naziv_vrste.



Slika 3.2: Reducirani Chenov dijagram baze podataka

U nastavku prikazujemo listu veza s atributima:

- spada_u, 1:N veza između tipova KATEGORIJA i PROIZVOD, s tim da PROIZVOD ima obavezno članstvo.
- izrađuje, 1:N veza između tipova PROIZVOĐAČ i PROIZVOD, s tim da PROIZVOD ima obavezno članstvo.
- uključuje, N:M veza između tipova PROIZVOD i NARUDŽBA, s atributima količina, cijena, datum_isporuke, isporučeno_dn.
- upućuje, 1:N veza između tipova KUPAC i NARUDŽBA, s tim NARUDŽBA ima obavezno članstvo.
- plaća_je, 1:N veza između tipova KREDITNA KARTICA i NARUDŽBA, s tim da NARUDŽBA ima obavezno članstvo.

- posjeduje, 1:N veza između tipova KUPAC i KREDITNA KARTICA, s tim da KREDITNA KARTICA ima obavezno članstvo.
- pripada, 1:N veza između tipova VRSTA KARTICE i KREDITNA KARTICA.

3.2 Logičko oblikovanje baze podataka

Druga faza oblikovanja baze podataka je logičko oblikovanje. Cilj ove faze oblikovanja je stvoriti relacijsku shemu baze, odnosno shemu koja će opisati logičku strukturu baze u skladu s pravilima relacijskog modela podataka. U relacijskoj shemi nalaze se entiteti i veze među njima koje su pretvorene u relacije.

Građa relacije kratko se opisuje shemom tablice. To je redak koji se sastoji od imena tablice te popisa imena atributa odvojenih zarezima i zatvorenih u zagrade. Obično je primarni ključ podvučen kako bi se lakše shvatila relacijska shema. Relacijska shema za primjer koji će se prikazati bi onda izgledala ovako:

- PROIZVOĐAČ (id_proizvođača, naziv_tvrtke, adresa_tvrtke, kontakt, email)
- KATEGORIJA (id_kategorije, naziv_kategorije)
- PROIZVOD (id_proizvoda, id_proizvođača, id_kategorije, naziv, slika, broj_jedinica, cijena_po_komadu, raspoloživ_dn)
- KUPAC (id_kupca, titula, ime, prezime, adresa, kontakt, email)
- KREDITNA KARTICA (id_kartice, id_kupca, id_vrste, datum_isteka, broj_kartice)
- VRSTA KARTICE (id_vrste, naziv_vrste)
- NARUDŽBA (id_narudžbe, id_kupca, id_kartice, datum_primitka, vrijeme_primitka, datum_obrade, izvršena_dn)
- UKLJUČUJE (id_proizvoda, id_narudžbe, količina, cijena, datum_isporuke, isporučeno_dn)

3.3 Fizičko oblikovanje baze podataka

Zadnja faza podatkovnog modeliranja predstavlja kreiranje fizičkog podatkovnog modela. Kao sustava za upravljanje bazom podataka (DBMS) ćemo koristiti Microsof SQL Server.

Microsoft SQL Server sustav je za upravljanje relacijskim bazama podataka koji je razvila tvrtka Microsoft. Radi se o softverskom proizvodu kojem je primarna funkcija

pohranjivanje i dohvaćanje podataka prema zahtjevima drugih softverskih aplikacija koje se mogu pokretati na jednom ili više računala preko mreže. Microsoft je izdao preko deset različitih verzija Microsoft SQL Servera namijenjenih različitim korisnicima pojedinačnih aplikacija do velikih aplikacija s velikim brojem korisnika.

Prva verzija SQL Servera koji ima veze sa Microsoftom (ranije je Sybase proizvodio SQL server pod imenom „Sybase SQL server“) izašla je 1989. godine pod imenom „SQL Server for OS/2 1.0“. Ta verzija je bila identična Sybase-ovom SQL serveru 3.0 koji je radio pod Unix sistemom. Microsoft SQL Server se pod tim imenom počeo prodavati 1992. godine, a puno ime mu je glasilo Microsoft SQL Server 4.2. Prvi SQL Server za Windows NT izašao je isto kada i sami Windows-i. U tom trenutku Sybase i Microsoft su se razišli, te je Sybase počeo razvijati svoju bazu podataka pod imenom Adaptive Server Enterprise. Do 1994. Microsoft je svejedno morao na sada svojem sustavu nositi Sybaseovu oznaku zbog autorskih prava.

Od trenutka raskida sa Sybaseom, Microsoft je napravio značajne pomake u razvoju svoje baze podataka. SQL Server je prva baza podataka na svijetu koja je posjedovala korisničko sučelje. Sve baze su u tadašnje vrijeme radili pomoću „command-line“ sistema koji zna biti izrazito nezgodan i nezgrapan. Također, SQL Server je prva komercijalna baza podataka koja je podržala Intelovu 64-bitnu arhitekturu procesora.

Posljednje izdana verzija je Microsoft SQL Server 2019 koja donosi inovativne značajke u području sigurnosti i usklađenosti te naprednije mogućnosti podatkovne analitike s podrškom za obradu velikih podataka. SQL Server 2019 može raditi s petabajtom podataka, pri čemu je znatno poboljšana obrada podataka te izvođenje upita iz različitih vanjskih izvora podataka, kao što su SQL Server, Oracle, Teradata te MongoDB. Izlaskom Microsoft SQL Servera 2017 omogućeno je njegovo korištenje na različitim platformama kao što su Windows, Linux te Docker kontejneri pa je ova značajka unaprijeđena i na novoj verziji SQL Servera 2019. Kada je riječ o performansama i sigurnosti, SQL Server je jedan od vodećih RDBMS sustava te pripada kategoriji najmanje ranjivih baza podataka. Sve dobre značajke replicirane su na noviju inačicu SQL Servera 2019.

U tablici 3.1 su prikazane cijene za Microsoft SQL Server ovisno o vrsti izdanja.

Tablica 3.1: Cijene Microsoft SQL Servera ovisno o vrsti izdanja

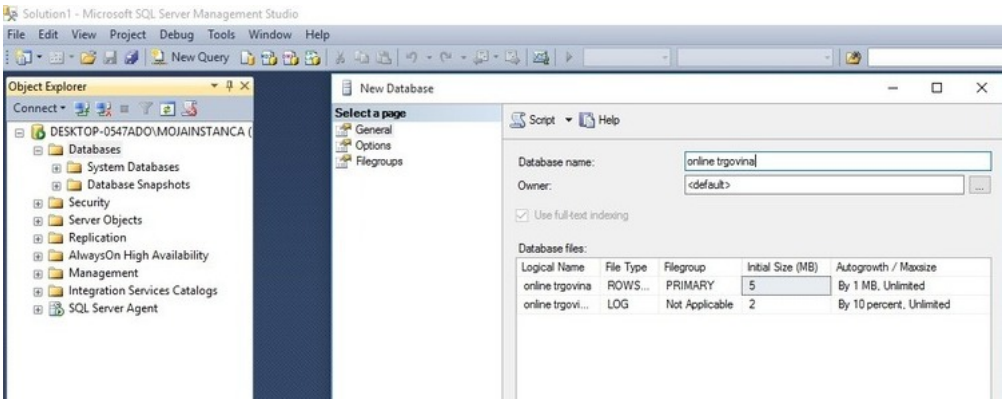
SQL Server izdanje	Idealno za	cijena
Enterprise	Idealno za sveobuhvatne, kritične performanse za zahtjevne baze podataka i zahtjeve poslovne inteligencije.	$14,256
Standard - per core	Idealno za mogućnosti upravljanja temeljnim podacima i poslovnom inteligencijom za nekritično radno opterećenje s minimalnim IT resursima.	$3,717
Standard - server + CAL	Idealno za mogućnosti upravljanja temeljnim podacima i poslovnom inteligencijom za nekritično radno opterećenje s minimalnim IT resursima.	$931
Developer	Potpuno opremljena verzija SQL Server softvera koja omogućuje programerima da isplativo grade, testiraju i demonstriraju aplikacije temeljene na SQL Server softveru.	Besplatno
Web	Sigurna, ekonomična i visoko skalabilna podatkovna platforma za javne web stranice. Dostupno samo davateljima softverskih usluga trećih strana.	-
Express	Besplatna baza podataka koja je idealna za učenje, kao i za izradu aplikacija na radnom stolu i malih poslužitelja do 10 GB.	Besplatno

U ovom poglavlju ćemo prikazati kako se kreira baza podataka i neki njezini objekti pomoću SQL Server Management Studio (SSMS) alata. SQL Server Management Studio (SSMS) integrirano je okruženje za upravljanje bilo kojom SQL infrastrukturom, od

SQL Servera do Azure SQL baze podataka. SSMS nudi alate za konfiguriranje, nadgledanje i administraciju primjeraka SQL Servera i baza podataka. SSMS se koristi za implementaciju, nadgledanje i nadogradnju komponenata podatkovnog sloja kojeg koriste vaše aplikacije i za izradu upita i skripti. SSMS se također koristi za ispitivanje, oblikovanje i upravljanje bazama podataka i skladištima podataka.

3.3.1 Baza podataka

Bazu podataka moguće je kreirati izvršavanjem SQL (DDL) upita ili pomoću SQL Server Management Studio (SSMS) alata gdje se u konačnici opet generiraju SQL upiti čijim se izvršavanjem kreira zadana baza podataka. Upotrebom SSMS alata, baza podataka kreira se na način da se desnim klikom na stavku ”Databases” u prikazanom izborniku odabere stavka ”New database”, a nakon toga se prikazuje obrazac prikazana na slici 3.3.



Slika 3.3: Kreiranje baze podataka pomoću SSMS alata

Koncept vlasništva se prije koristio u starijim verzijama SQL Servera kada još uvijek nisu postojale sheme u bazama podataka. Vlasnik je korisnik koji ima nepovratne ovlasti nad pojedinim objektom ili bazom podataka, a njega nije moguće izbrisati sve dok sva svoja vlasništva ne prebaci na drugog korisnika. Međutim danas se objekti najčešće grupiraju po shemama što olakšava administraciju korisnika i dozvola, te zbog toga nije potrebno definirati vlasnika baze već se u polje Owner stavlja vrijednost ”default”.

Još ćemo spomenuti neke postavke pri izradi baze podataka. Postavkom ”Collation” se definiraju pravila pri sortiranju podataka s različitim tipovima znakova, gdje se vrijednost ”default” ne treba mijenjati jer SQL Server podržava Unicode tipove znakova.

Postavka ”Recovery model” izravno utječe na rad dnevnika transakcija te način izrade i povrata rezervnih kopija. Podržani modeli oporavka baze podataka su ”full”, ”bulk-logged” i ”simple”.

Kao preporuka i najčešće korišteni model oporavka koristi se ”full” model. U dnevniku transakcija spremaju se podaci o svakoj izvršenoj transakciji, pa ga je moguće koristiti pri povratu baze podataka u slučaju greške te u slučaju povrata u neku prijašnju točku u vremenu. Nedostatak korištenja ovog modela su velike datoteke dnevnika transakcija u zahtjevnim sustavima s velikim brojem transakcija.

Model oporavka ”bulk-logged” najčešće se koristi tek privremeno u slučajevima izvršavanja velikog broja transakcija odjednom, dok se model oporavka ”simple” rijetko koristi, samo u testnim okruženjima, jer se iz datoteke dnevnika transakcija automatski brišu sve završene transakcije i zbog toga dnevnik transakcija ne može biti više korišten za povrat baze podataka.

Korištenje SSMS alata pri kreiranju baze podataka prikladno je zbog automatskog generiranja potrebnih SQL naredbi i zbog cijelog niza postavki koje bi se inače morale pamtiti i ručno podešavati.

3.3.2 Tablice baze podataka

Tablice su tipovi objekata koji služe za spremanje podataka. Sastoje se od redaka i stupaca, gdje jedan redak sadrži jedan zapis. Dijelovi pojedinog zapisa identificirani su stupcima koji imaju svoje ime i tip, a dodatno mogu biti opisani ograničenjima i svojstvima. Tablice se kreiraju izvršavanjem DDL SQL upita.

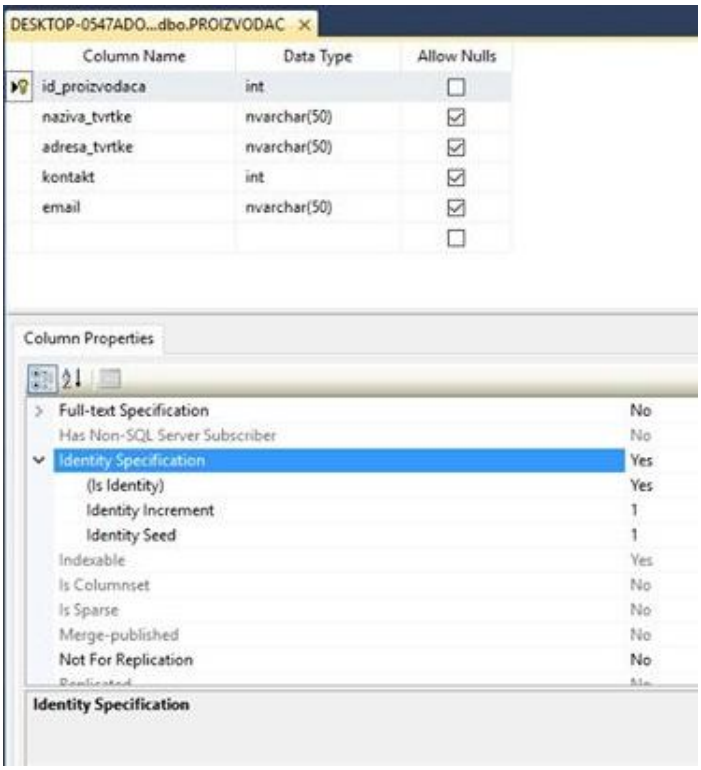
SQL Server Management Studio (SSMS) pruža mogućnost jednostavnog kreiranja i uređivanja tablica bilo korištenjem interaktivnog grafičkog sučelja ili izravnim izvršavanjem SQL upita, jer će pristup preko interaktivnog grafičkog sučelja rezultirati pozadinskim generiranjem i izvršavanjem odgovarajućih SQL upita.

Pri kreiranju tablice uvijek treba paziti na postojanost primarnog ključa. Iako nije nužno da tablica mora imati primarni ključ njegovo nepostojanje može narušiti konzistentnost baze podataka (pojava duplikata) te vrlo često može ukazivati na pogrešno oblikovanje i biti uzrokom loših performansi pri pretraživanju i spajanju s drugim tablicama. Ukoliko u tablici nema stupca koji bi mogao biti primarni ključ, onda se dodaje cjelobrojni stupac sa svojstvom identiteta čija se vrijednost automatski povećava svaki put kada se dodaje novi zapis, a u našem slučaju to je *id_proizvodaca*.

Kreiranje tablice *PROIZVODAC* i njezinog primarnog ključa prikazano je na slici 3.4. Cjelobrojni stupac *id_proizvodaca* ima svojstvo identiteta (”Identity Specification = Yes”). Vrijednost ovog stupca za prvi zapis u tablici definirana je izrazom „Identity Seed”, a za svaki sljedeći zapis njegova vrijednost uvećava se za broj definiran izrazom ”Identity Increment”. Tako će u ovom slučaju vrijednost stupca *id_proizvodaca* započeti od 1, te će se za svaki sljedeći zapis uvećavati za 1.

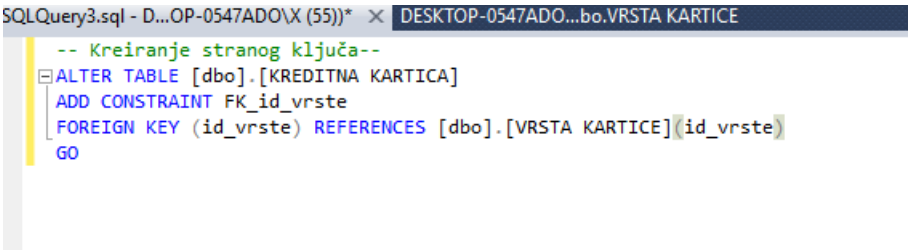
U svrhu očuvanja konzistentnosti i integriteta baze podataka u svakoj tablici moguće je definirati pravila za podatke. Ona se implementiraju kao ograničenja (*constraints*) nad

pojedinim stupcima tablice. Ograničenja su objekti baze podataka koji mogu biti tipa PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, DEFAULT, INDEX i CHECK.



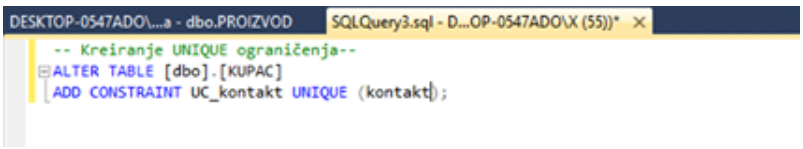
Slika 3.4: Kreiranje tablice i postavljanje vrijednosti primarnog ključa

FOREIGN KEY je ključ koji se koristi za povezivanje dviju tablica. FOREIGN KEY je polje (ili zbirka polja) u jednoj tablici koja se odnosi na PRIMARY KEY u drugoj tablici. Na primjeru sa slike 3.5 kreiran je strani ključ za tablicu *KREDITNA KARTICA* i stupac *id_vrste*.



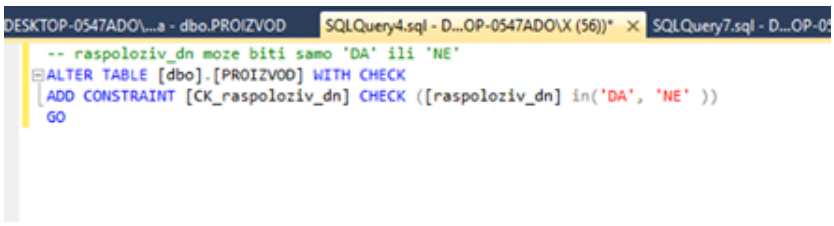
Slika 3.5: Kreiranje stranog ključa

UNIQUE ograničenje osigurava da su sve vrijednosti u stupcu različite. UNIQUE i PRIMARY KEY ograničenja jamče jedinstvenost za stupac ili skup stupaca. Ograničenje PRIMARY KEY automatski ima UNIQUE ograničenje. Međutim, po tablici se može imati više UNIQUE ograničenja, ali samo jedno PRIMARY KEY ograničenje. Za tablicu *KUPAC* i stupac *kontakt* dodano je UNIGUE ograničenje, a to je prikazano na slici 3.6.

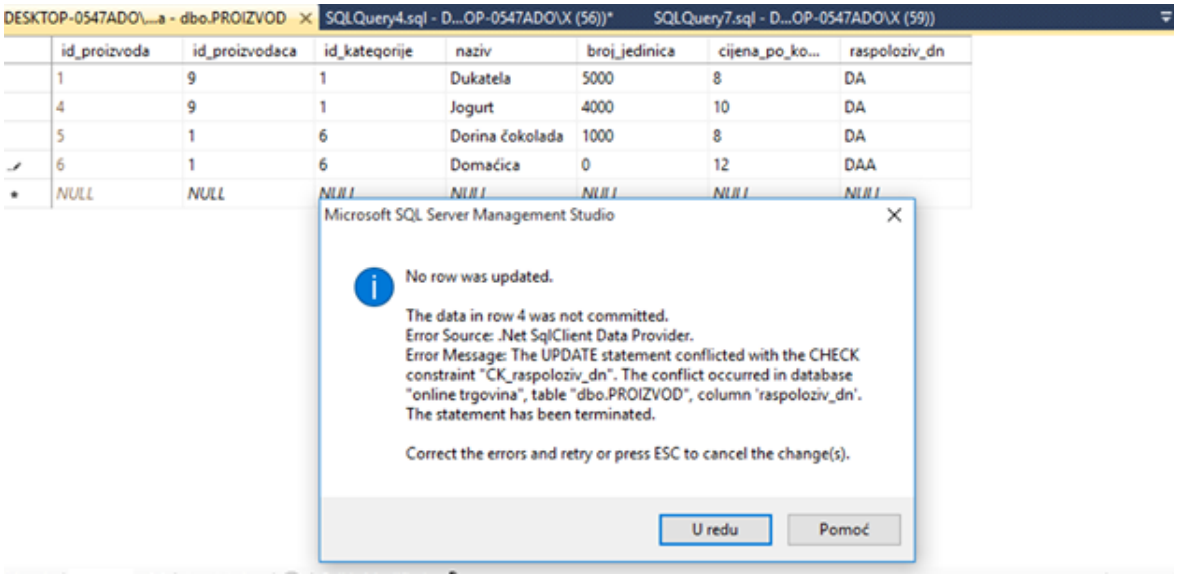


Slika 3.6: Kreiranje UNIQUE ograničenja

Ako je potrebno provjeravati intervale vrijednosti i karakteristike podataka (kao što je minimalna i maksimalna duljina) u pojedinim stupcima koristi se ograničenje tipa CHECK. CHRECK ograničenje prikazano je na slici 3.7 gdje smo u tablici *PROIZVOD* ograničili stupac *raspoloziv_dn* na način da taj prima vrijednost ”DA” ili ”NE”. Ako probamo unijeti neku drugu vrijednost javit će se greška kao što je prikazano na slici 3.8.



Slika 3.7: Kreiranje CHECK ograničenja

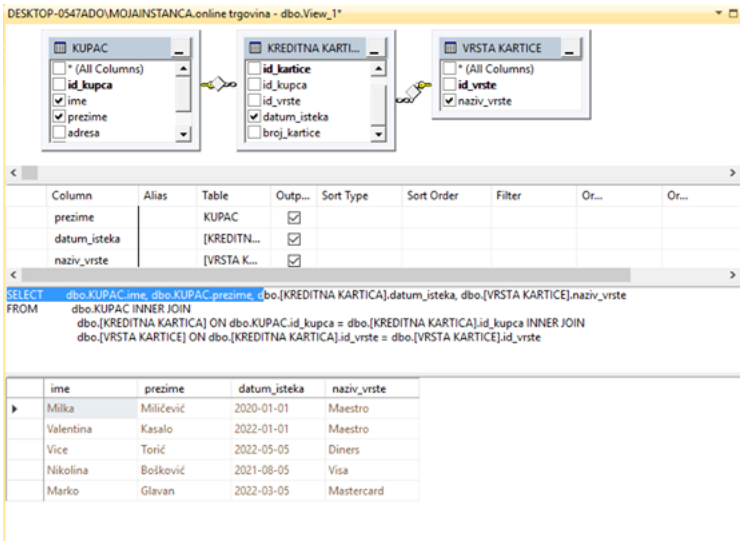


Slika 3.8: Javljanje greške za krivi unos podataka

3.3.3 Pogledi

Pogled je objekt baze podataka koji sadrži tekst spremljenog SELECT upita. Koristi se kako bismo SELECT upitom mogli povezati podatke iz više različitih tablica te ih na pojednostavljen način prikazati i koristiti kao da su smješteni u samo jednoj (virtualnoj) tablici. Pogled (njegov SELECT upit) se ne može izvršiti sam od sebe, već se pri njegovom referenciranju unutar nekog drugog SQL upita ili pogleda stvara rezultantna virtualna tablica.

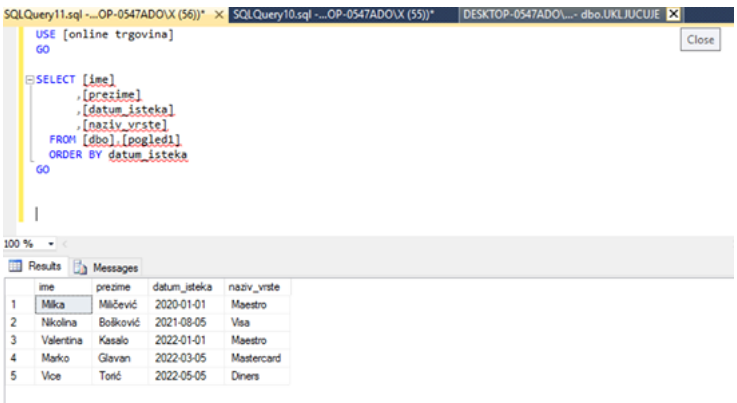
Poglede je moguće kreirati korištenjem SSMS alata ili izravnim izvršavanjem SQL upita. Na slici 3.9 prikazano je kreiranje pogleda pomoću SSMS alata.



Slika 3.9: Kreiranje pogleda

U pogledu se ispisuju svi kupci, vrsta kreditne kartice koju posjeduju i datum isteka kartice. U prvom koraku izrade pogleda se prvo biraju tablice, koje su međusobno povezane pomoću stranih i primarnih ključeva što je prikazano i na slici. Nakon toga je potrebno odabrati stupce koji će biti vidljivi kao rezultat izvršavanja pogleda. Odabirom tablica i stupaca automatski se generira i nadopunjuje odgovarajući SQL upit pogleda koji je također prikazan na slici. I na kraju, izvršavanjem pogleda kreira se virtualna tablica koja je prikazana na slici.

Nakon što je pogled kreiran, moguće ga je koristiti u SQL upitima, što je prikazano na slici 3.10. Pogledi imaju više ograničenja, a jedno od njih je i nepouzdano korištenje izraza ORDER BY. Sortiranje podataka se ne radi u definicij pogleda, nego se to radi korištenjem izraza ORDER BY pri referenciranju pogleda iz nekog drugog SQL upita.



Slika 3.10: Upotreba pogleda

Unutar pogleda se ne mogu stvarati nove tablice, koristiti varijable i privremene tablice, ne može se kreirati pogled s parametrima, a to su samo neki od nedostataka pogleda. Tablicu pogleda je vrlo često moguće ažurirati iako je onda virtualna. Takvi ažurirajući pogledi zapravo ažuriraju tablice na osnovu kojih je nastao rezultat pogleda.

Pogledi nude neke sigurnosne prednosti unatoč svim nedostatcima. Unutar pogleda moguće je sakriti informacije o svim referentnim tablicama, shemama i drugim korištenim pogledima, a korisniku dati dozvolu samo za čitanje podataka dobivenih pogledom.

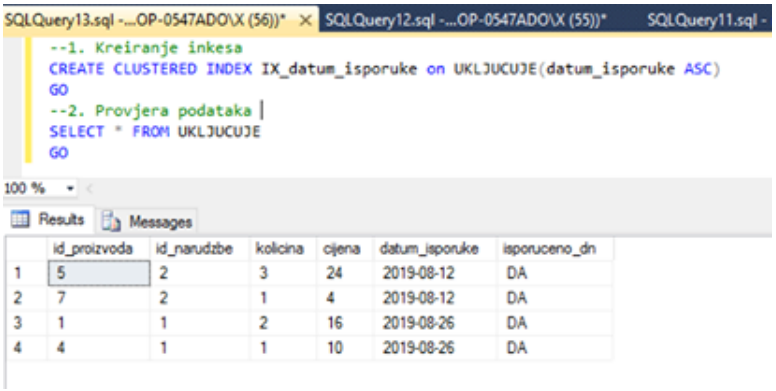
3.3.4 Indeksi

Veće količine podataka u tablicama i pogledima usporavaju izvršavanje SQL upita zbog sporije pretrage i dohvata tih podataka. Primjerice, pretraga nekog podatka u skupu od par milijuna zapisa u najgorem slučaju može značiti sekvencijalni pregled svih zapisa sve dok se ne pronađe traženi podatak. Takav način pretrage naziva se skeniranje te najčešće predstavlja "najskuplji" način pretrage u smislu potrošnje procesorskog vremena. Da bismo ga izbjegli koristimo indekse. Najčešći tipovi indeksa su grupirajući (*clustered*) i negrupirajući (*non-clustered*) indeksi.

Grupirajući indeks možemo zamisliti kao telefonski imenik u kojem su svi telefonski brojevi poredani po prezimenu. Binarnom pretragom lagano je pronaći broj neke osoba na osnovi njenog prezimena. Problem bi nastao ako bi telefonski imenik htjeli pretražiti po imenu osobe, onda bismo morali sekvencijalno pregledavati svaki pojedini zapis.

Grupirajućim indeksom definira se fizički razmještaj zapisa u tablici, a on može biti samo jedan u zadano vrijeme. Na primjer, u isto vrijeme ne možemo imati zapise u tablici (telefonskom imeniku) sortirane po imenu i po prezimenu jer bi za to trebali imati dvije različite tablice. Iz tog razloga u tablici može postojati samo jedan grupirajući indeks. Ako se treba stvoriti drugi, onda se prethodno prvi mora izbrisati.

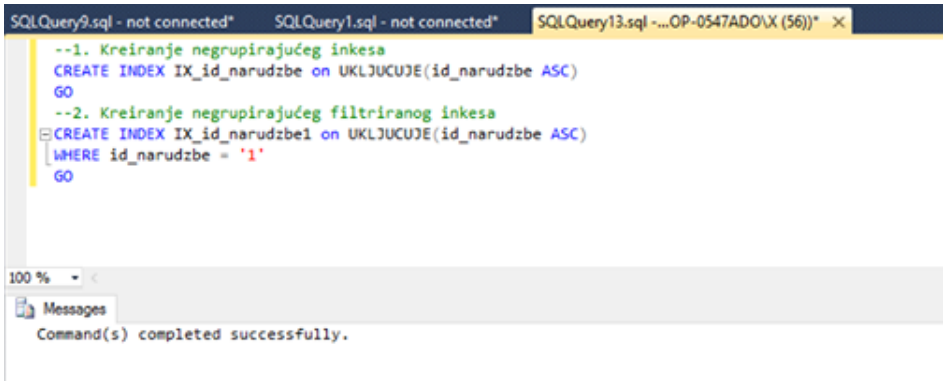
U našoj bazi podataka grupirajući indeks možemo napraviti samo na tablici *UKLJUCUJE*, jer jedino ona nema zadan primarni ključ. Na slici 3.11 prikazano je kreiranje grupirajućeg indeksa, gdje smo podatke grupirali po datumu isporuke. Nakon kreiranja indeksa IX_datum svi zapisi u tablici *UKLJUCUJE* fizički su sortirani po vrijednosti stupca *datum_isporuke*, što se također može vidjeti na slici.



Slika 3.11: Kreiranje indeksa

Kreiranjem svakog novog indeksa vrijednost performanse pada pri operacijama dodavanja, izmjene i brisanja podataka, jer nakon ovih operacija najčešće će trebati ažurirati i sve postojeće indekse, a ta ažuriranja mogu oduzeti mnogo procesorskog vremena. Iz tih razloga indekse treba pažljivo odabrati kako ne bi uzrokovali lošiju performansu.

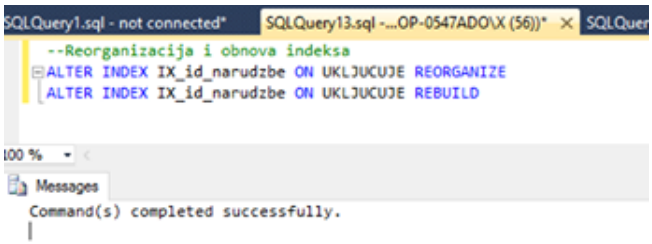
Negrupirajući indeks kreira se na vrlo sličan način kao i grupirajući indeks. SQL Server omogućava i kreiranje filtriranih negrupirajućih indeksa. Njih se koristi u slučajevima kada se učestalo rade pretrage određenog podskupa podataka. Kreiranje negrupirajućeg i filtriranog negrupirajućeg indeksa je prikazano na slici 3.12.



Slika 3.12: Kreiranje negrupirajućeg i filtriranog negrupirajućeg indeksa

Česte promjene na podacima (dodavanje, izmjena i brisanje) mogu uzrokovati fragmentaciju indeksa, pa indeksne stanice postanu razbacane ili ne budu dobro popunjene, a to loše utječe na performansu. SQL Server tada nudi mogućnost reorganizacije ili obnove indeksa, što je prikazano na slici 3.13. Ako je fragmentacija indeksa manja od 30%

onda se preporuča reorganizacija indeksa, a reorganizacija indeksa podrazumijeva reorganizaciju indeksnih stranica. Ako je fragmentacija indeksa veća od 30% onda se preporuča obnova indeksa tj. njihovo brisanje i ponovno pokretanje. Obnova indeksa može dugo trajati što ovisi o količini podataka u tablici, pa ju je poželjno raditi kada je broj upita prema bazi najmanji.



Slika 3.13: Reorganizacija i obnova indeksa

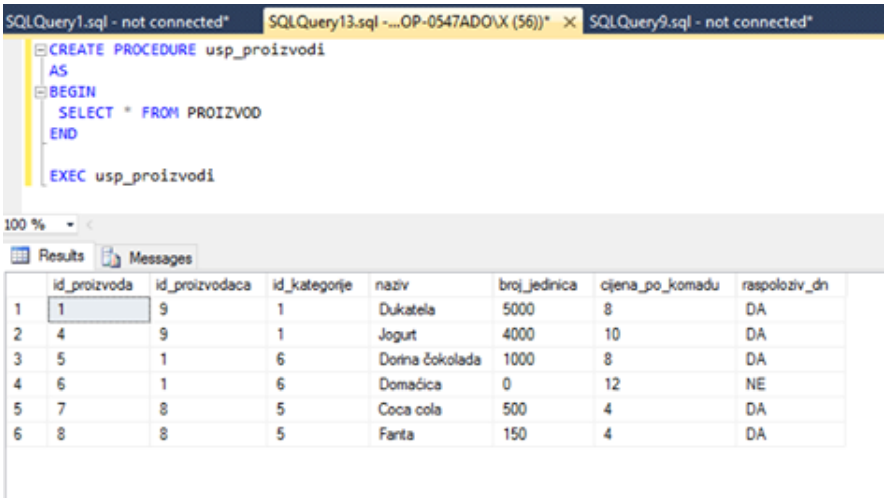
3.3.5 Uskladištene procedure i funkcije

Korisnički definiranim procedurama i funkcijama moguće je centralizirati i nametnuti istu poslovnu logiku svim klijentima baze podataka. Tako možemo osigurati da svi klijenti koriste iste rutine pri obradi podataka. Ovakvim pristupom postiže se konzistentnost u radu svih klijenata.

SQL Server nudi mogućnost pisanja vlastitih uskladištenih procedura, skalarnih funkcija i funkcija s tabličnim vrijednostima.

Uskladištene procedure omogućuju spremanje SQL upita u bazi podataka. Nema više potrebe za slanjem kompleksnih upita mrežom već je dovoljno pozvati uskladištenu proceduru koja sadrži odgovarajući upit. One se izvršavaju na serveru, što doprinosi performansama. Uskladištene procedure omogućuju i dodatne razine sigurnosti jer korisnik ne mora imati ovlasti za rad s objektima, niti ovlasti za izvršavanje upita koji se nalaze u uskladištenoj proceduri, ali može biti ovlašten izvršiti uskladištenu proceduru koja sve te objekte i upite sadrži.

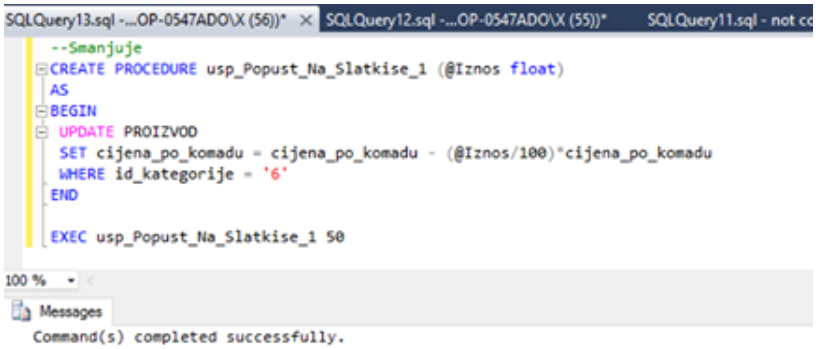
Tijelo uskladištene procedure omeđeno je ključnim riječima BEGIN i END. Ono može sadržavati više naredbi, a izvršava se naredbom EXECUTE. Kreiranje procedure je prikazano na slici 3.14.



	id_proizvoda	id_proizvodaca	id_kategorije	naziv	broj_jedinica	cijena_po_komadu	raspoloziv_dn
1	1	9	1	Dukatela	5000	8	DA
2	4	9	1	Jogurt	4000	10	DA
3	5	1	6	Dorina čokolada	1000	8	DA
4	6	1	6	Domaćica	0	12	NE
5	7	8	5	Coca cola	500	4	DA
6	8	8	5	Fanta	150	4	DA

Slika 3.14: Kreiranje procedure

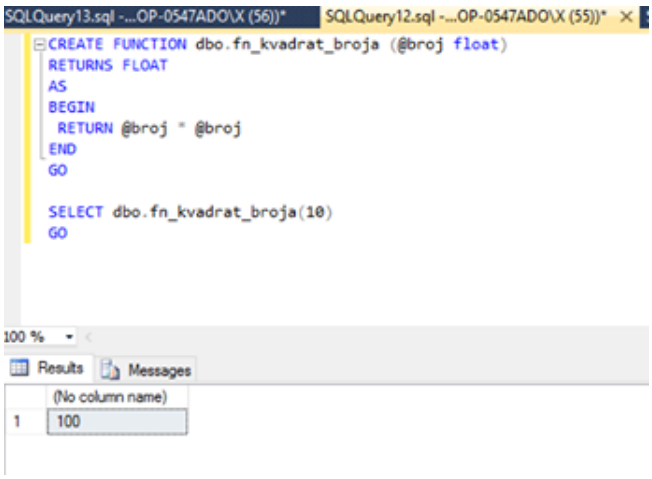
Uskladištene procedure mogu mijenjati sadržaj baze podataka. Tako procedura *usp_Popust_Na_Slatkise_1* smanjuje cijenu slatkiša za 50% (Slika 3.15).



Slika 3.15: Kreiranje procedure

Uskladištena procedura može vratiti set podataka, cjelobrojnu vrijednost korištenjem naredbe RETURN ili vrijednosti drugih tipova korištenjem izlaznih (eng. output) tipova parametara.

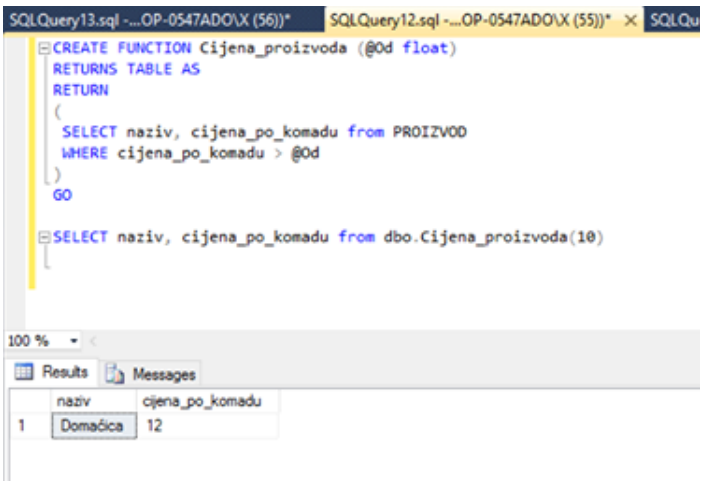
Skalarne funkcije moraju vratiti samo jednu vrijednost koja se definira naredbom RETURN. Skalarne funkcije za iste vrijednosti ulaznih parametara uvijek moraju dati istu povratnu vrijednost. Na slici 3.16 je prikazan primjer kreiranja i poziva skalarne funkcije.



Slika 3.16: Kreiranje skalarne funkcije i njezin poziv

Također, postoje dva tipa funkcija koje vraćaju setove podataka, odnosno tablične vrijednosti. To su: Inline Table-Valued funkcije i Multistatement Table-Valued funkcije.

Prvi tip predstavlja jednostavni oblik funkcije s tabličnom vrijednošću. Tijelo sadrži jednu SELECT naredbu kojom se definira povratna vrijednost funkcije, tj. rezultantna virtualna tablica. Znog toga je slična pogledu, ali za razliku od pogleda može imati parametre kao i svaka druga funkcija dok ih pogled nema. Na slici 3.17 je prikazano kreiranje funkcije *Cijena_proizvoda* koja vraća virtualnu tablicu s popisom proizvoda čija je cijena veća od 10kn, njezin poziv i rezultat poziva.



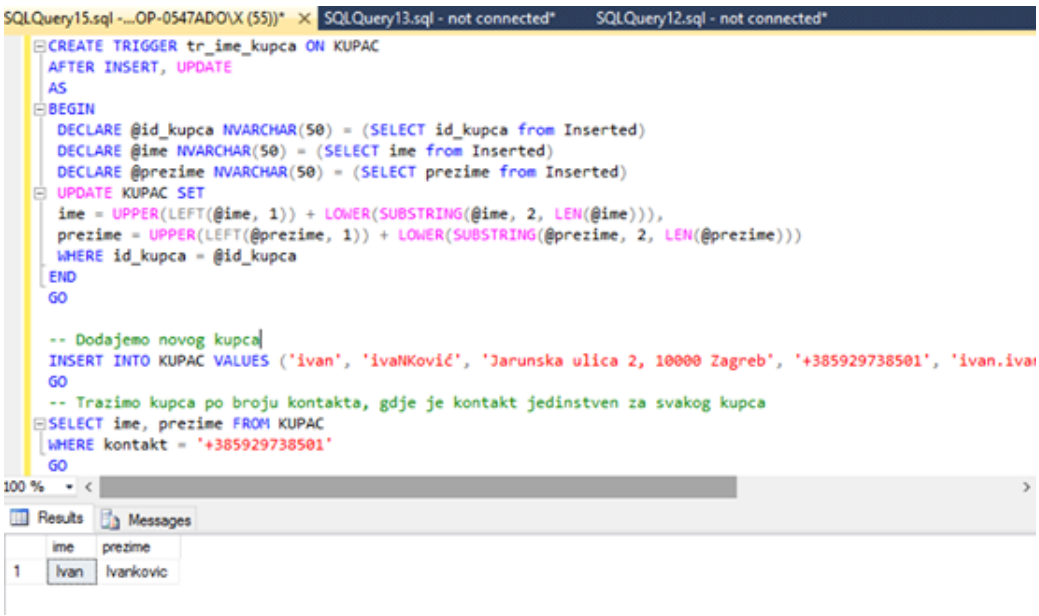
Slika 3.17: Jednostavni oblik funkcije sa tabličnom vrijednošću

Ako je potrebno da funkcija s tabličnom vrijednošću sadrži više naredbi, onda koristimo složeni oblik. Kod njega je potrebno definirati strukturu rezultantne tablice. Blok naredbi mora biti omeđen ključnim riječima BEGIN i END. Sadržaj rezultantne tablice se vraća naredbom RETURN.

3.3.6 Okidači

Okidač (*trigger*) je posebna vrsta pohranjene procedure koja se automatski pokreće kad se događaj dogodi na poslužitelju baze podataka. Postoje tri vrste okidača, a to su: DML, DDL i Logon okidači. Najčešće se koriste DML i DDL okidači. U SSMS alatu DML okidače moguće je pronaći unutar svake tablice pod stavkom "Triggers", DDL okidače na razini baze podataka pod stavkom "Database Triggers", a okidače na razini servera u stavci "Server Objects/Triggers".

DML okidači pokreću se kada korisnik pokušava izmijeniti podatke događajem jezika manipulacije podacima (DML). DML događaji su izrazi INSERT, UPDATE ili DELETE na tablici ili pogledu. Oni se aktiviraju kada se aktivira bilo koji valjani događaj, bilo da su pogođeni redovi tablice ili ne. Primjer DML okidača nakon operacija INSERT i UPDATE prikazan je na slici 3.18.

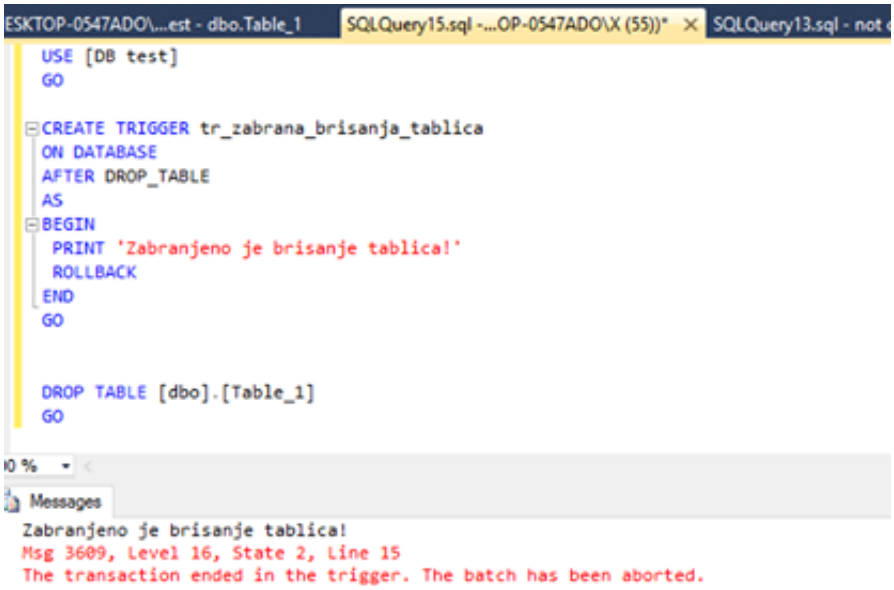


Slika 3.18: Kreiranje DML okidača nakon operacije INSERT i UPDATE i njegovo testiranje

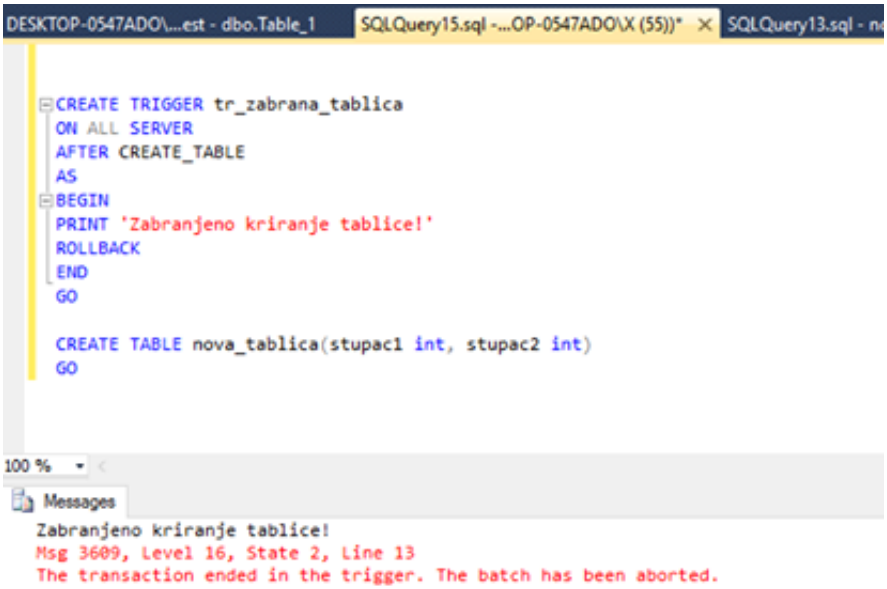
Okidačem *tr_ime_kupca* je implementirano poslovno pravilo gdje se traži da se prilikom dodavanja novog kupca ili ažuriranja postojećeg ime i prezime kupca počinje velikim slovom (ostala slova su mala).

Okidač se izvršava jednom po operaciji, a ne jednom po svakom zahvaćenom zapisu (retku), pa je zbog toga potreban oprez pri operacijama gdje se zahvaća više zapisa odjednom. Okidač sa slike 3.18 nije moguće izvršavati nad više zahvaćenih zapisa odjednom, već samo nad INSERT i UPDATE operacijama u kojima se zahvaća po jedan zapis.

Za kontrolu strukture objekata koristimo DDL okidače. DDL okidači pokreću se kao odgovor na različite događaje jezika definiranja podataka (DDL). Ovi događaji prvenstveno odgovaraju izrazima CREATE, ALTER i DROP i određenim sistemski pohranjenim procedurama koje izvode operacije slične DDL-u. DDL okidač može djelovati na razini baze (ON DATABASE) i na razini servera (ON ALL SERVER). Na slici 3.19 je prikazano kreiranje DDL okidača na razini baze podataka, dok je na slici 3.20 prikazano kreiranje na razini servera.

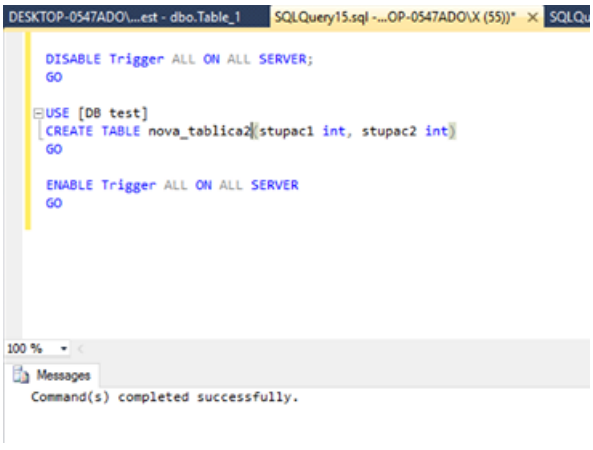


Slika 3.19: Kreiranje DDL okidača na razini baze podataka



Slika 3.20: Kreiranje DDL okidača na razini servera

DDL okidač sa slike 3.19 zabranjuje brisanje tablica, dok je s DDL okidačem sa slike 3.20 zabranjeno kreiranje tablica u bilo kojoj bazi podataka na serveru. Zbog ovakvih i sličnih situacija DML i DDL okidače privremeno je moguće onemogućiti, pa izvršiti potrebne operacije, onda zatim ponovno omogućiti što je prikazano na slici 3.21.



Slika 3.21: Onemogućavanje i omogucavanje okidača na razini servera

3.4 Organizacija i sigurnost

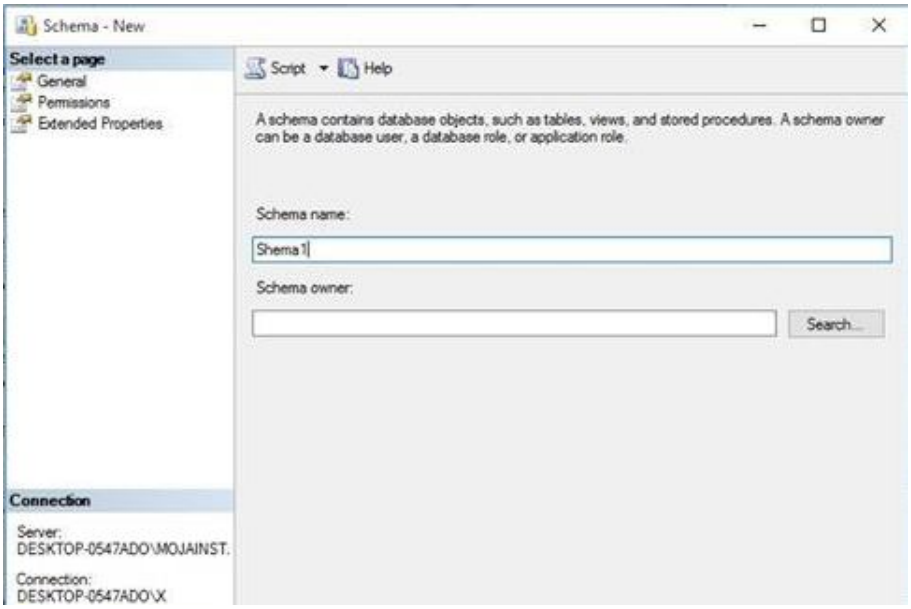
3.4.1 Sheme

U starijim verzijama SQL Servera svaki objekt morao je imati svog vlasnika. Vlasnik objekta najčešće bi bio korisnik koji je kreirao objekt i imao je trajne (nepovratne) ovlasti nad tim objektom. Problemi bi nastali kada bismo htjeli izbrisati korisnika iz baze podataka jer tada bismo sve objekte u njegovom vlasništvu prethodno trebali prebaciti u vlasništvo nekog drugog korisnika.

Izlaskom SQL Servera 2005 uvode se sheme koje uvelike pojednostavljuju organizaciju, administraciju i pristup objektima baze podataka. Shemu možemo zamisliti kao imenovani prostor (*namespace*) u kojemu se nalazi jedan ili više objekata. Svaki objekt baze podataka može pripadati samo jednoj shemi (podrazumijevano dbo), a referencira ga se upravo preko imena sheme kojoj pripada (Server.BazaPodataka.Shema.Objekt).

Sheme omogućuju organizaciju objekata baze podataka po namjeni, sadržaju ili nekom drugom kriteriju. Sheme mogu sadržavati tablice, poglede, uskladištene procedure, funkcije itd. Baza podataka može sadržavati i više objekata s istim imenom, a u tom slučaju svaki od tih objekata mora biti smješten u različitoj shemi.

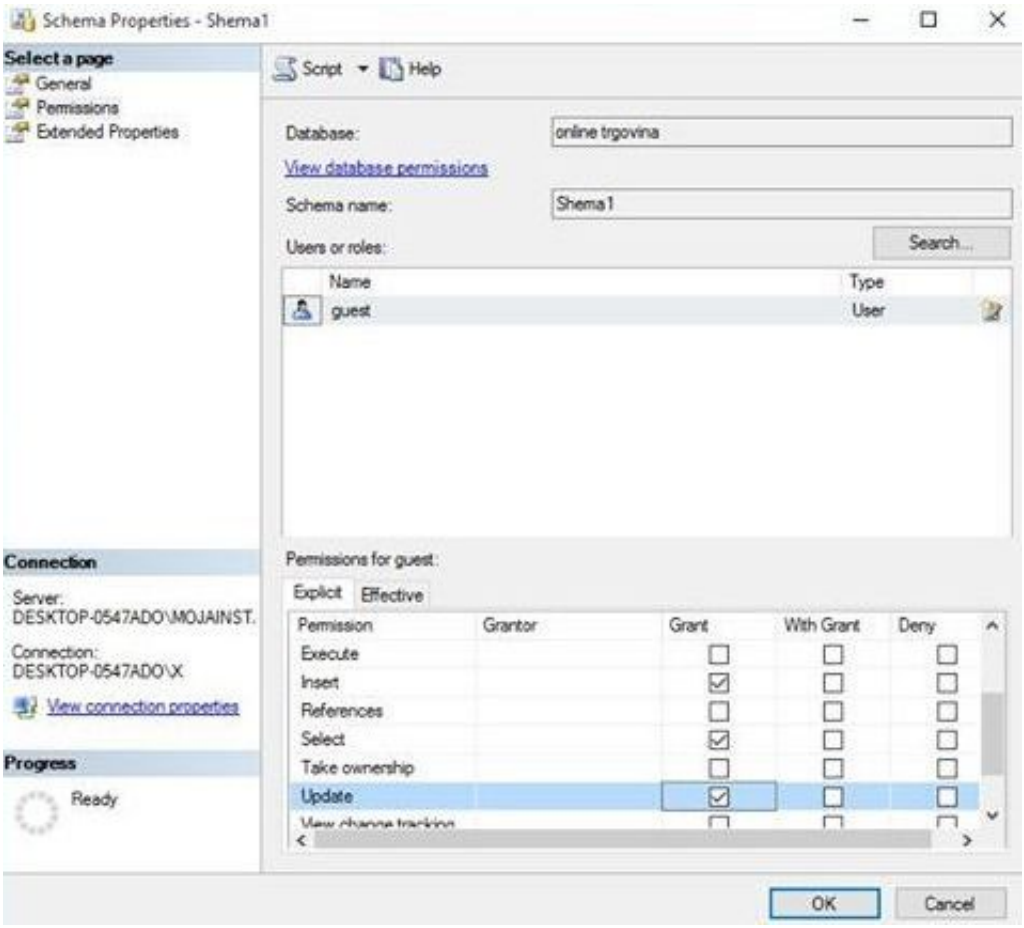
Upotrebom SSMS alata moguće je kreirati novu shemu i pregledati sve do tad kreirane sheme. Na slici 3.22 je prikazana izrada sheme.



Slika 3.22: Izrada sheme

Svaka shema ima vlasnika, a ako ga ne definiramo, onda se podrazumijeva da korisnik dbo postaje vlasnik. Zbog toga je ovakav pristup najčešći. U protivnom, za sve ostale korisnike koji su vlasnici treba voditi računa o prijenosu vlasništva pri pokušaju njihovog brisanja iz baze podataka. Jedna od prednosti shema je i jednostavnija administracija. Umjesto da korisniku definiramo ovlasti nad svakim objektom pojedinačno, možemo mu definirati ovlasti nad shemom. Tada te ovlasti automatski ima i za sve objekte unutar te sheme.

Na slici 3.23 imamo primjer sheme gdje korisnik *guest* ima ovlasi za dodavanje, mijenjanje i dohvaćanje podataka iz svih objekata unutar sheme, dok se iste ovlasti mogu definirati i za skupine korisnika (*database role*) te aplikacije (*application role*). Korisnicima je moguće odobriti (GRANT) ili zabraniti (DENY) neku operaciju, a opcijom WITH GRANT omogućiti da tu operaciju odobre i drugim korisnicima.

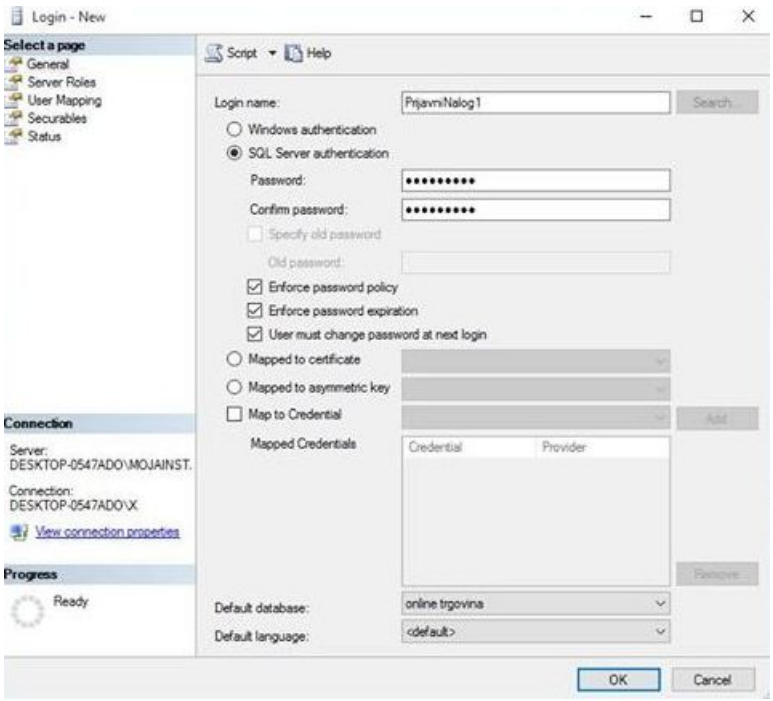


Slika 3.23: Izrada sheme - ovlasti

3.4.2 Prijavni nalozi

Prijavni nalog koristi se u svrhu autentifikacije i spajanja korisnika na SQL Server. Svaki prijavni nalog može biti povezan s više korisničkih računa baza podataka (jednim korisničkim računom po bazi), a oni se koriste za autorizaciju, odnosno pristup bazama podataka.

Prijavni nalog kreira se na razini instance, a moguće ga je kreirati korištenjem SSMS alata ili uporabom SQL upita. Na slici 3.24 prikazana je izrada prijavnog naloga. Prilikom izrade prijavnog naloga vidimo da možemo odabrati Windows autentifikaciju ili SQL Server autentifikaciju. Odabirom Windows autentifikacije nije potrebno definirati lozinku i ovaj tip autentifikacije nije pogodan za korisnike koji se nalaze u drugim domenama ili pristupaju SQL Serveru preko interneta.



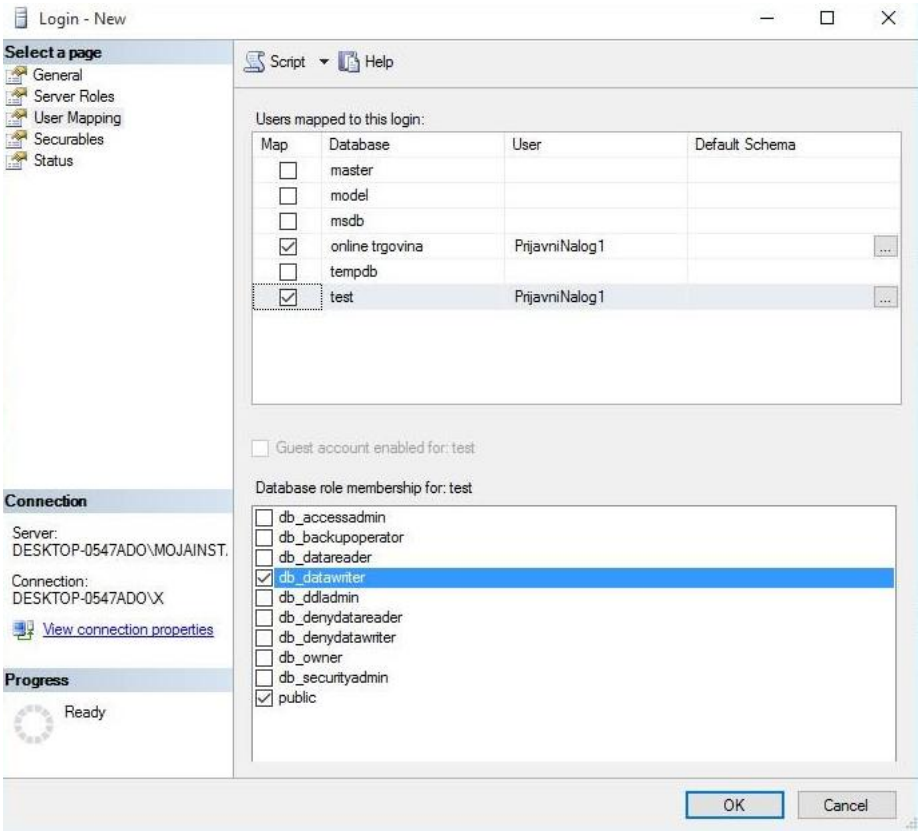
Slika 3.24: Izrada prijavnog naloga

SQL Server autentifikacija se koristi za podršku najšireg spektra korisnika. Podaci korisnika, tj. korisničko ime i lozinka spremaju se u SQL Serveru. Ti korisnici imaju mogućnost autentificiranja i spajanja na SQL Server iz drugih domena, mreža ili preko interneta, a moguće im je nametnuti i pravila o lozinkama, poput da lozinka traje samo određeno vrijeme ili da ju korisnik mora promijeniti pri sljedećoj prijavi. Pravila o lozinkama ne definiraju se u SQL Serveru već u operacijskom sustavu server računala.

Svakom prijavnom nalogu moguće je definirati podrazumijevanu bazu podataka i jezik. Ukoliko odabrana podrazumijevana baza podataka nije dostupna (izbrisana je ili sl.) autentifikacija neće biti moguća.

Prijavni nalog može imati više server uloga (*server roles*). Njih možemo zamisliti kao skupine korisnika koji imaju određene ovlasti na razini SQL Servera. Neke od uloga su: *sysadmin* uloga, a to je uloga čiji članovi nemaju nikakvih ograničenja i mogu raditi bilo što u SQL Serveru, *dbcreator* uloga koja omogućava korisnicima kreiranje novih baza, *securityadmin* uloga s kojom se može kontrolirati sigurnost na SQL Serveru, *processadmin*, *diskadmin*, *bulkadmin* i mnoge druge.

Tijekom kreiranja prijavnog naloga moguće je kreiranje i povezivanje novih korisničkih računa baza podataka. Na slici 3.25 je prikazano povezivanje prijavnog naloga *PrijavniNalog1* s bazama podataka *online trgovina* i *test* na način da se unutar njih kreiraju korisnički računi koji imaju isto ime kao i prijavni nalog.



Slika 3.25: Kreiranje i povezivanje korisničkih računa baza podataka

Svaki od ovih korisničkih računa može imati različito ime te može pripadati različitim ulogama baza podataka (*database roles*). I u ovom slučaju postoje unaprijed definirane

neke uloge poput *db_datareader*, *db_datawritter*, *db_owner* i tako dalje. U našem primjeru će se u bazama podataka *online trgovina* i *test* kreirati novi korisnički račun *PrijavniNalog1* koji će pripadati *db_datawriter* ulozi, tj. imati će ovlasti pisanja podataka u bazu podataka.

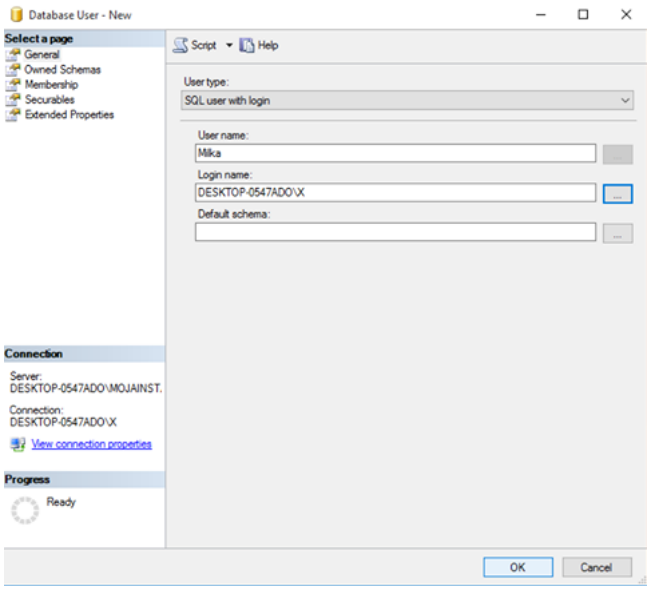
Prijavnom nalogu moguće je dodijeliti ovlasti na razini servera (stavke "Securables" i "Status"). Primjerice, to mogu biti ovlasti za spajanje i gašenje servera, pregled, kreiranje i izmjenu baza podataka, izmjenu prijavnih naloga itd.

3.4.3 Korisnički računi

Korisničkim računima vrši se autorizacija, odnosno dodjeljivanje prava i dozvola za rad u bazi podataka. Najčešće se koriste u kombinaciji s prijavnim nalozima gdje se nakon uspješne autentifikacije, ovisno o povezanim korisničkim računima, korisniku dodjeljuju prava i dozvole nad jednom ili više baza podataka. Korisnički računi smješteni su unutar baze podataka.

SSMS alat će ponuditi izradu korisničkog računa baze podataka za sljedeće tipove korisnika: Windows korisnik, SQL korisnik s prijavnim nalogom, SQL korisnik bez prijavnog naloga, korisnik povezan s certifikatom i korisnik povezan s asimetričnim ključem.

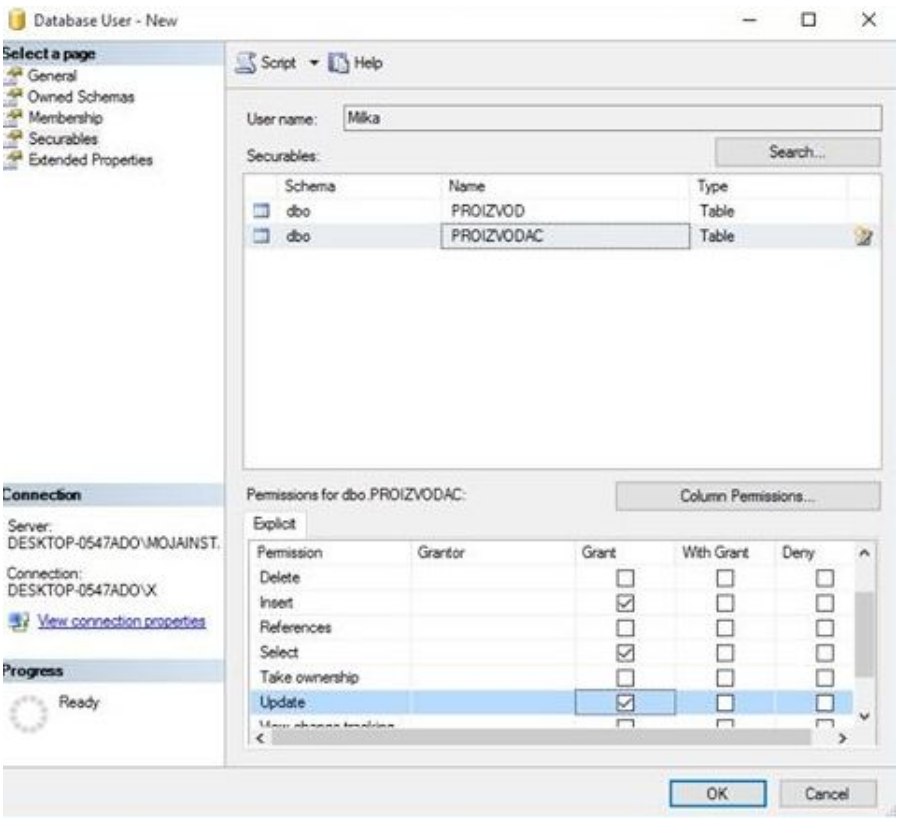
Ukoliko korisnički račun baze podataka nije kreiran prilikom kreiranja prijavnog naloga, moguće ga je kreirati naknadno upotrebom SQL upita ili u SSMS alatu. Izrada korisničkog računa je prikazana na slici 3.26.



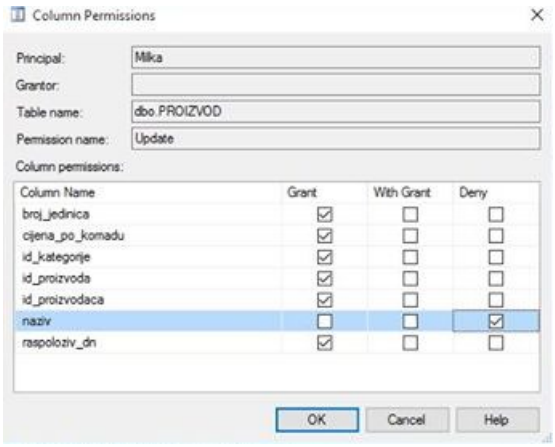
Slika 3.26: Kreiranje i povezivanje korisničkih računa baza podataka

Najčešće će korisnički račun baze podataka biti kreiran na osnovu postojećeg Windows korisnika ili će biti riječ o SQL korisniku s prijavnim nalogom. Oba tipa korisničkih računa povezuju se s prijavnim nalogom, a mogu biti vlasnici jedne ili više shema te članovi jedne ili više uloga baze podataka što se može birat u stavkama "Owned Schemas" i "Membership".

Prava i dozvole u bazi podataka, dodjeljuju se unutar stavke "Securables". U našem primjeru na slici 3.27 korisniku *Milka* će biti dozvoljeno izvršavanje INSERT, SELECT i UPDATE upita nad tablicama *PROIZVOD* i *PROIZVODAC*. Kako je riječ o tablicama možemo tu dozvolu dodijeliti na razini pojedinog stupca klikom na gumb "Column Permissions...". Klikom na taj gumb se prikazuje obrazac koja je prikazana na slici 3.28. Korisniku *Milka* je dozvoljeno izvršavanje INSERT, SELECT i UPDATE upita nad tablicama *PROIZVOD* i *PROIZVODAC*, ali ta dozvola se neće odnositi na stupac *naziv* u tablici *PROIZVOD*.



Slika 3.27: Dodjela dozvola korisničkom računu baze podataka



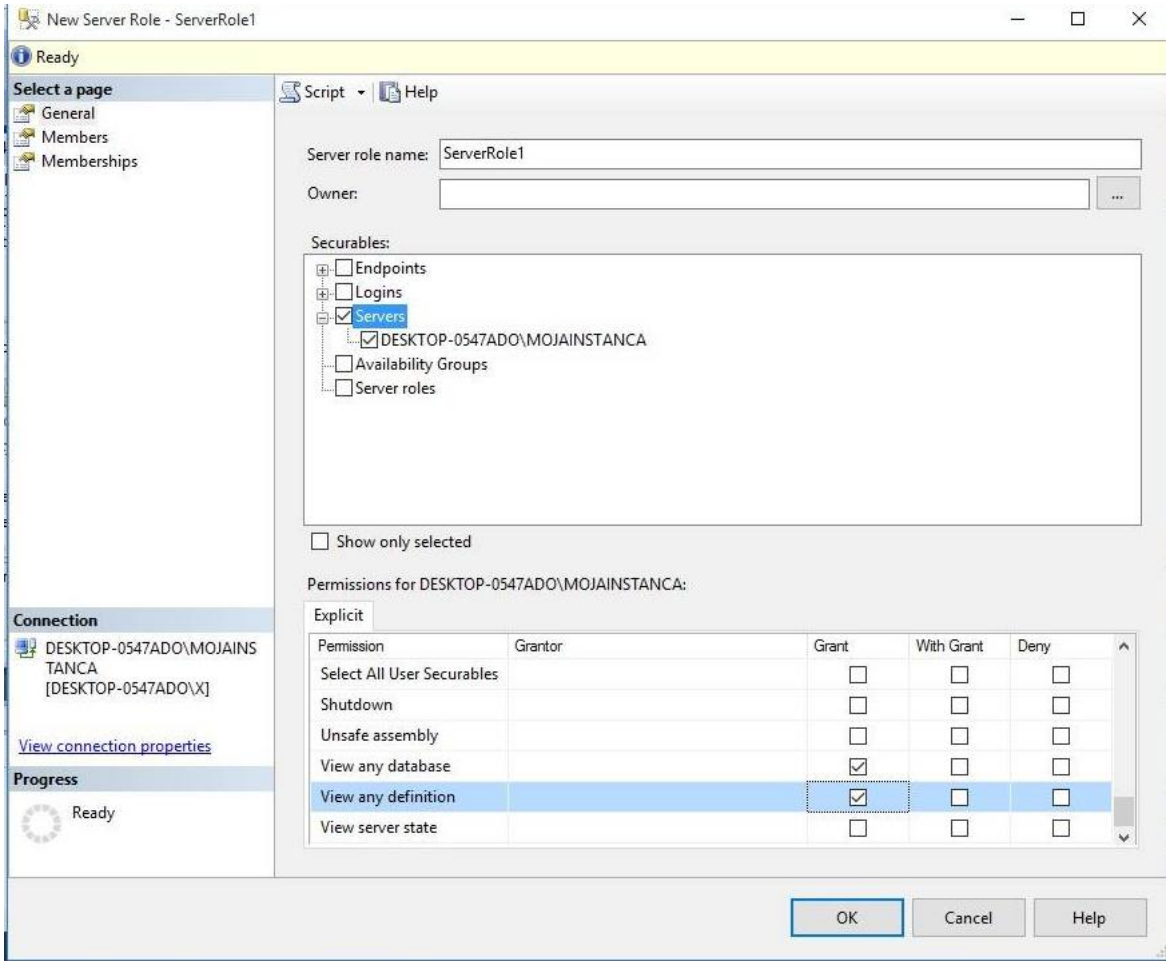
Slika 3.28: Dodjela dozvola korisničkom računu baze podataka

Od korisnika posebne namjene vrlo često se upotrebljava SQL korisnik bez prijavnog naloga (*SQL user without login*). Najčešće se koristi kada je riječ o autorizaciji klijent aplikacija i ovaj tip korisničkog računa nema lozinku.

3.4.4 Uloge

Uloge se najčešće koriste za grupiranje korisnika. Pomoću uloga jednostavnije je organizirati i administrirati korisnike jer im se prava i dozvole mogu automatski dodijeliti na osnovi članstva u nekoj ulozi. U SQL Serveru moguće je kreirati tri tipa uloga: serverske uloge, uloge baze podataka, aplikacijske uloge.

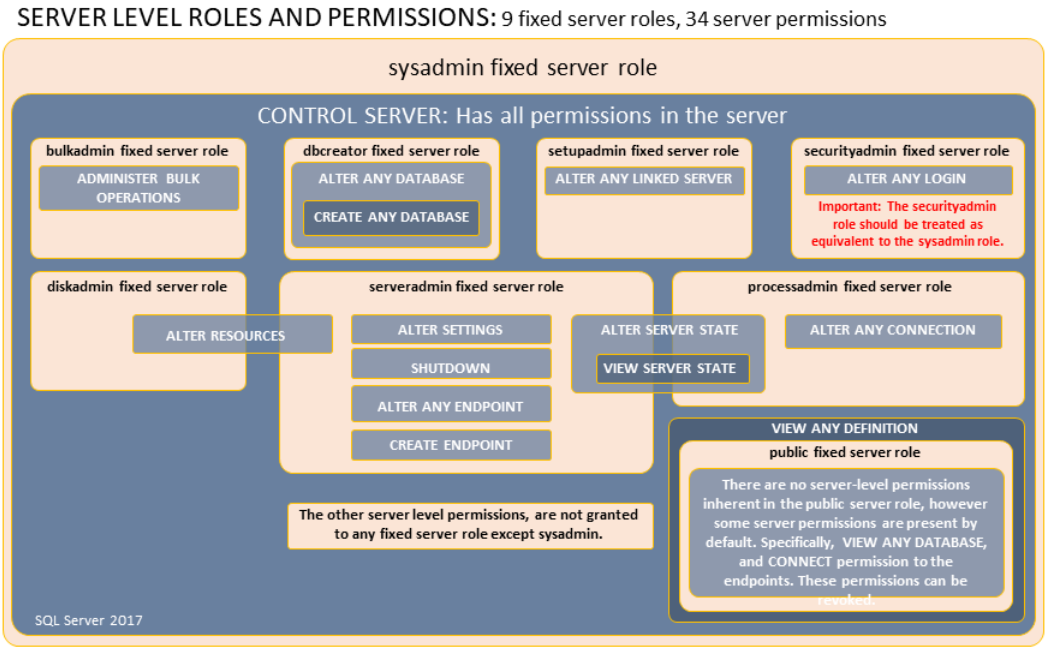
Serverskom ulogom definira se skupina članova koja ima prava i dozvole na razini SQL Server instance, a članovi serverske uloge mogu biti prijavni nalozi i druge serverske uloge. U primjeru na slici 3.29 svi korisnici kojima je dodjeljena uloga ”ServerRole1” će imati prava i dozvole za pregled popisa svih baza podataka (*View any database*) i definicije objekata (*View any definition*). Server ulogom moguće je dodijeliti prava i dozvole za upravljanje pristupnim točkama, prijavnim nalozima itd.



Slika 3.29: Kreiranje server uloge ”ServerRole1”

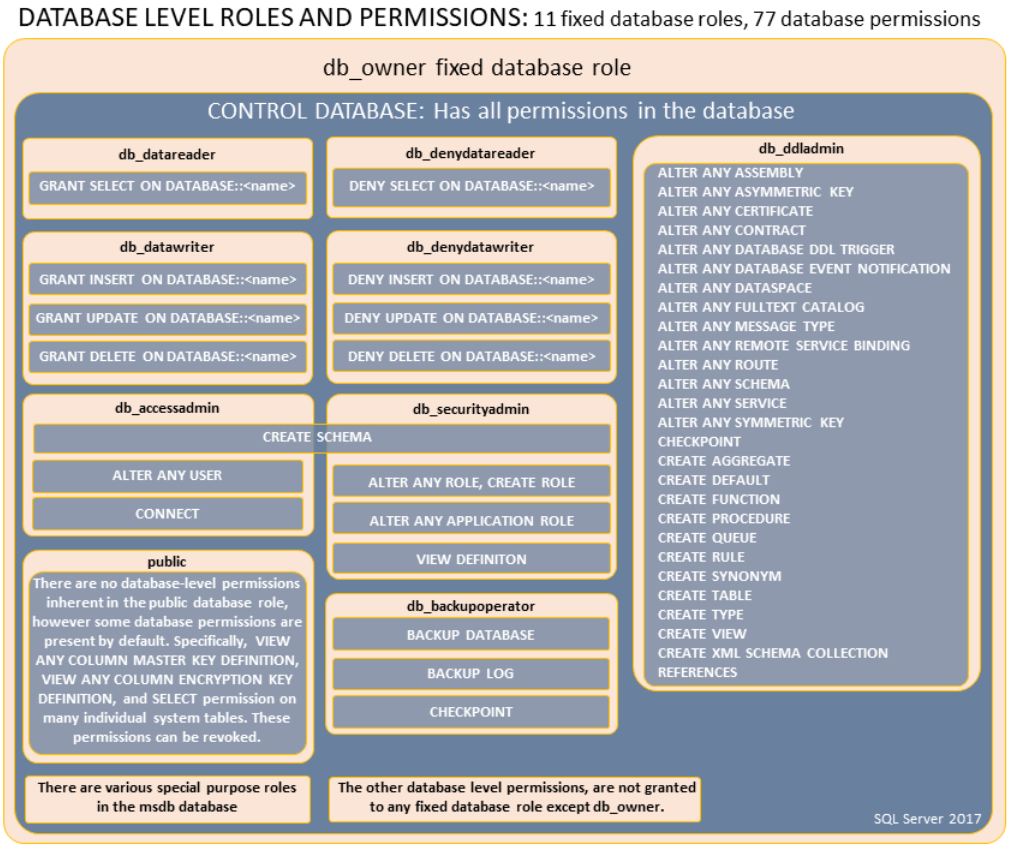
Pri kreiranju nove ili uređivanju postojeće serverske uloge, moguće joj je dodijeliti nove članove odabirom stavke ”Members”. Također, ukoliko je potrebno da navedena serverska uloga bude član neke druge serverske uloge (njena poduloga) to se može definirati u stavci ”Membership”.

Svaka SQL Server instanca ima već unaprijed definiran niz stalnih serverskih uloga koje se mogu iskoristiti pri dodjeljivanju najčešćih administrativnih prava i dozvola. Stalne serverske uloge ne mogu se mijenjati već im je tek moguće kontrolirati članstvo. Na slici 3.30 je prikazan popis i prava stalnih serverskih uloga.



Slika 3.30: Popis stalnih serverskih uloga i njihova prava

Uloge baze podataka se koriste za grupiranje i lakšu administraciju korisničkih računa baze podataka. Osim korisničkih računa, članovi te uloge mogu biti i druge (pod)uloge baze podataka. Kao i kod serverskih uloga i tu imamo stalne uloge baze podataka koje su prikazane na slici 3.31, a ukoliko korisnik nije prikladan za članstvo niti u jednoj od tih uloga, može se kreirati nova ili se prava i dozvole tom korisniku mogu dodijeliti eksplicitno, izvan uloge.



Slika 3.31: Popis stalnih uloga baze podataka i njihova prava

Baza podataka može sadržavati i aplikacijske uloge, ali one za razliku od serverskih uloga i uloga baze podataka ne služe za grupiranje određenog tipa korisnika već za autorizaciju klijent aplikacija. Svaka aplikacija trebala bi se autorizirati preko jedne aplikacijske uloge, pri čemu aplikacija mora znati naziv uloge i njenu lozinku.

Povezivanje s aplikacijskom ulogom je dano u sljedećim koracima:

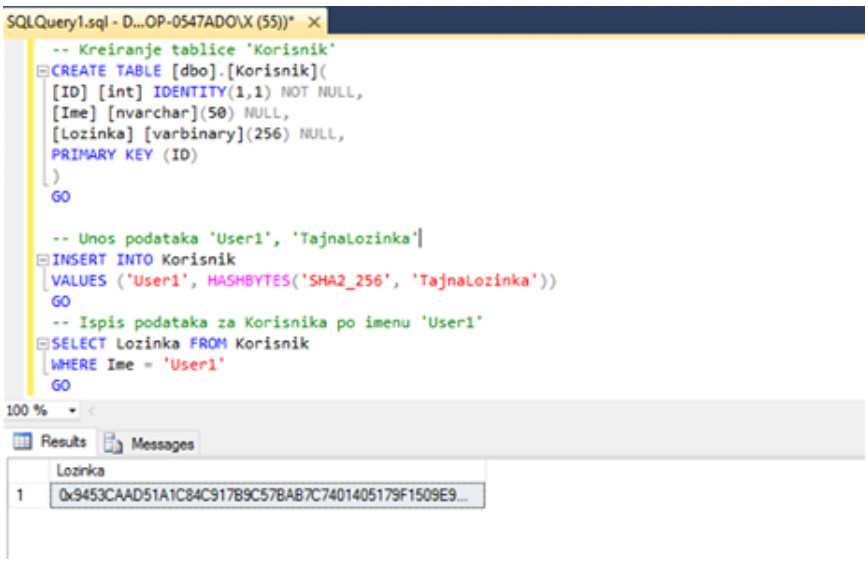
- Korisnik izvršava aplikaciju za klijenta.
- Klijentska aplikacija povezuje se s instancom SQL Servera kao korisnika.
- Aplikacija zatim izvršava pohranjenu proceduru sp_setapprole s lozinkom koja je poznata samo aplikaciji.
- Ako su naziv i lozinka aplikacijske uloge važeći, aplikacijska uloga je omogućena.
- U tom trenutku veza gubi dopuštenja korisnika i preuzima dopuštenja aplikacijske uloge.

3.4.5 Kriptografija u SQL Serveru

Dodjeljivanjem odgovarajućih prava i dozvola moguće je ograničiti pristup podacima. Uljezi često uspijevaju doći do korisničkih podataka, pa u tom trenutku imaju pristup i potencijalno osjetljivim podacima koji se nalaze u bazi podataka (osobni podaci kupaca, lozinke, brojevi kreditnih kartica itd.). Zbog toga osjetljive podatke uvijek je poželjno dodatno zaštiti. U tu svrhu SQL Server omogućuje korištenje funkcija sažimanja, te simetričnih i asimetričnih kriptografskih algoritama.

Funkcije sažimanja koriste se kao jednosmjerni kriptografski algoritmi bez ključa. Kada se neki sadržaj (tekst, datoteka ili sl.) sažme funkcijom sažimanja više nije moguće iz sažetog sadržaja dobiti nazad izvorni sadržaj. Zbog toga se one često koriste za zaštitu lozinki. Rezultat funkcije sažimanja mora biti jedinstven te uvijek isti za pojedinu ulaznu vrijednost. Funkcije sažimanja uvijek vraćaju binarni zapis iste veličine.

SQL Server 2016 omogućuje korištenje funkcija MD2, MD4, MD5, SHA-0, SHA-1, SHA-2 (256) i SHA-2 (512). Sve osim SHA-2 funkcija smatraju se zastarjelima te će se pri njihovom korištenju javiti upozorenje. [7]



Slika 3.32: Zaštita lozinki korištenjem SHA2 – 256 funkcije

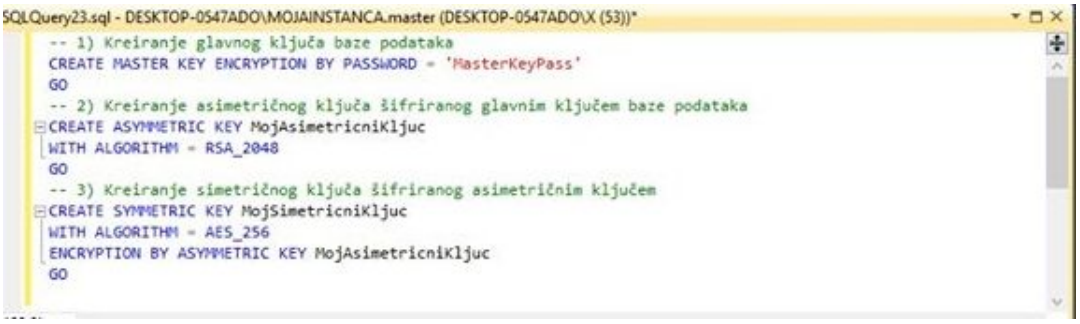
U primjeru smo kreirali tablicu *Korisnik* s atributima *ID*, *Ime* i *Lozinka*. U drugom SQL upitu smo unijeli podatke ”User1” i ”TajnaLozinka”. U trećem SQL upitu ispisujem lozinku za korisnika s imenom ”User1”, a kao rezultat vidimo da je to neki binarni zapis. Kada bi provjeravali lozinku, klijent aplikacija računa rezultat sažimanja unesene lozinke, te ga uspoređuje s rezultatom sažimanja koji je spremljen u bazi podataka. Ukoliko su oba

rezultata identična lozinka je ispravna. U protivnom korisnik će najčešće ponovno morati unijeti lozinku.

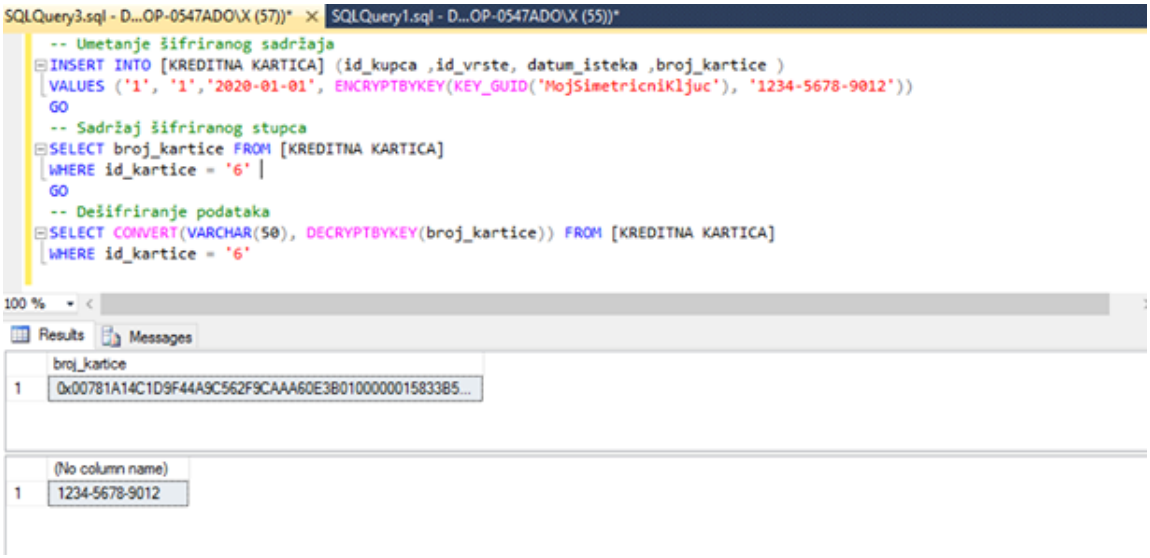
Funkcije sažimanja korisne su u situacijama gdje podatke nije potrebno vraćati u izvorni oblik, ali ako je to potrebno onda se koriste simetrični i asimetrični kriptografski algoritmi.

Simetrični kriptografski algoritmi koriste jedan tajni ključ. Jednom šifriran, sadržaj se može dešifrirati samo pomoću istog ključa. Ipak, da bi se tajni ključ na siguran način mogao dostaviti drugoj strani potrebno je koristiti asimetrične kriptografske algoritme koji koriste dva ključa (javni i privatni). Javni ključ svima je dostupan, a sadržaj šifriran javnim ključem može biti dešifriran samo odgovarajućim privatnim ključem kojeg posjeduje samo osoba kojoj je poruka namijenjena.

U bazi podataka online trgovine imamo tablicu *KREDITNA KARTICA* gdje imamo stupac *broj_kartice* kojeg trba šifrirati. Za šifriranje tog stupca koristit će se AES-256 algoritam pomoću simetričnog ključa "MojSimetricniKljuc". Da bi se zaštitio taj simetrični ključ kreiran je asimetrični ključ "MojAsimetricniKljuc" s algoritmom RSA (2048) čiji je privatni ključ također potrebno zaštiti. Podrazumijevano, štiti ga se pomoću glavnog ključa baze podataka (*database master key*, DMK) ili proizvoljno definiranom lozinkom. Kreiranje ključeva je prikazano na slici 3.33. Na slici 3.34 je prikazano kriptiranje i dekriptiranje broja kreditne kartice.



Slika 3.33: Kreiranje ključeva



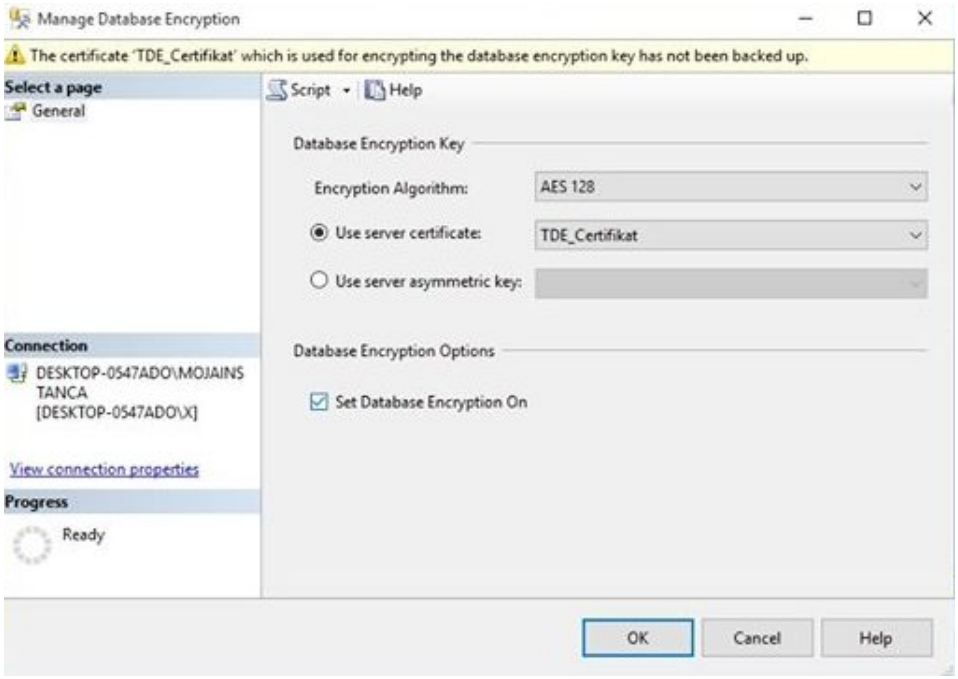
Slika 3.34: Korištenje simetričnog kriptografskog algoritma

3.4.6 Transparentno šifriranje podataka

Osim pojedinih stupaca, moguće je šifrirati i kompletnu bazu podataka korištenjem transparentnog šifriranja podataka (*transparent data encryption*, TDE). Cilj je zaštiti podatke u slučaju krađe pa se u tu svrhu šifriraju sve datoteke baze podataka. Za šifriranje sadržaja baze podataka i njenih datoteka koristi se simetrični algoritam čiji se ključ (*Database Encryption Key*, DEK) šifrira certifikatom smještenim u master bazi podataka ili asimetričnim ključem smještenim u EKM (*extensible key management*) modulu. Bez tog certifikata (ili asimetričnog ključa) uljez neće moći dešifrirati DEK pa time niti sadržaj baze podataka.

Na slici 3.35 je prikazno kreiranje DMK ključa i server certifikata, jer trebaju postojati u master bazi prije korištenja TDE-a. Tek nakon toga je moguće generirati DEK ključ i šifrirati bazu podataka. Prije korištenja TDE-a potrebno je se pobrinuti da u master bazi podataka postoji DMK ključ i certifikat koji će biti zaštićen tim ključem. Tek nakon toga moguće je generirati DEK ključ i šifrirati bazu podataka. Šifriranje baze podataka korištenjem TDE-a može se inicirati SSMS alatom tako da se desnim klikom na željenu bazu podataka odabere stavka ”Tasks / Manage Database Encryption”, a na slici 3.36 je prikazan obrazac koji se otvori.

Slika 3.35: Kreiranje DMK ključa i server certifikata



Slika 3.36: Šifriranje baze podataka (TDE) SSMS alatom

3.5 Održavanje i nadzor

3.5.1 Izrada rezervnih kopija

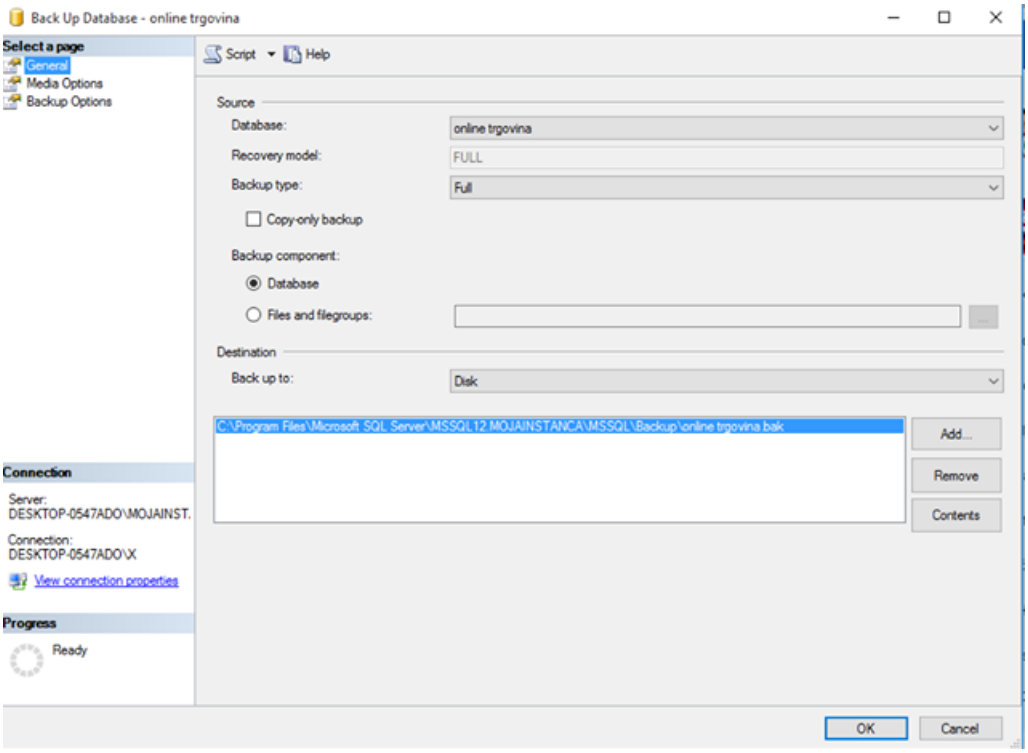
Da bi se smanjio rizik od gubitka podataka uslijed neočekivanih kvarova sklopovlja, pogrešnog rukovanja podacima ili u slučaju katastrofe poželjno je periodički izrađivati rezervne kopije baza podataka. Odabir odgovarajuće strategije za izradu i povrat rezervnih kopija jedna je od ključnih stavki kvalitetnog održavanja.

SQL Server podržava tri osnovna tipa rezervnih kopija, a to su: potpuna rezervna kopija (*full backup*), diferencijalna rezervna kopija (*differential backup*), rezervna kopija dnevnika transakcija (*transaction log backup*).

Potpuna rezervna kopija najsigurniji je oblik rezervne kopije, a njeno je postojanje i preduvjet za izradu diferencijalne kopije i kopije dnevnika transakcija.

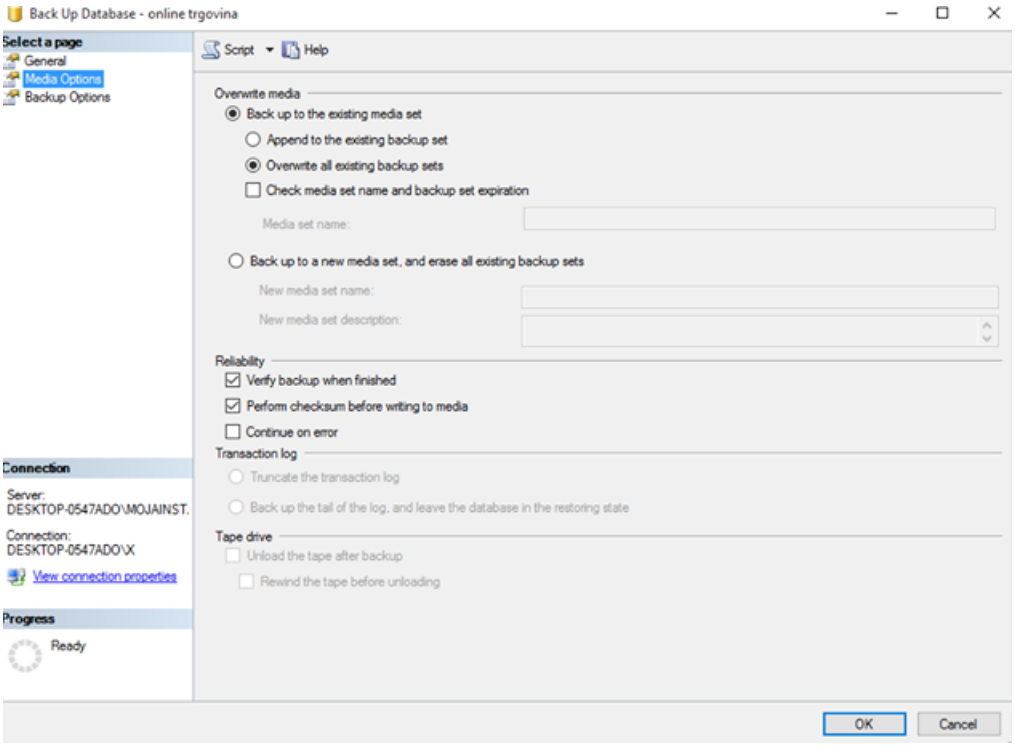
Na slici 3.37 prikazana je izrada potpune rezervne kopije upotrebom SSMS alata, a ona se može kreirati i izvršavanjem odgovarajućeg SQL upita. Vidimo da se može odabrati i stavka ”Copy-only Backup” koja se koristi u slučajevima kada se izrađuje rezervna kopija kojom se ne želi prekinuti već postojeći lanac rezervnih kopija. Tu stavku možemo koristiti za potrebe prebacivanja baze podataka u testno okruženje, jer izradom ovakvog tipa rezervne kopije ne utječe se na kasnije diferencijalne i rezervne kopije dnevnika transakcija. One neće kao polaznu točku koristiti kreiranu ”Copy-only Backup” rezervnu kopiju već tek zadnju potpunu rezervnu kopiju baze podataka.

Odredište rezervne kopije najčešće je datoteka na disku, a može biti i traka (*tape*) te URL. Svaki od ovih odredišta može biti zasebno kreiran i korišten kao uređaj rezervne kopije (*backup device*) koji postoji na razini SQL Server instance (stavka ”Server objects / Backup devices”). U primjeru na slici 3.37 odredište rezervne kopije je jedna datoteka na disku i to će često biti zadovoljavajući pristup jer će kompletan sadržaj rezervne kopije biti smješten samo na jednom mjestu. Međutim, kada je riječ o velikim bazama podataka izrada rezervnih kopija i njihov povrat može iziskivati puno vremena, pa se zbog toga za odredište bira više datoteka rezervnih kopija.



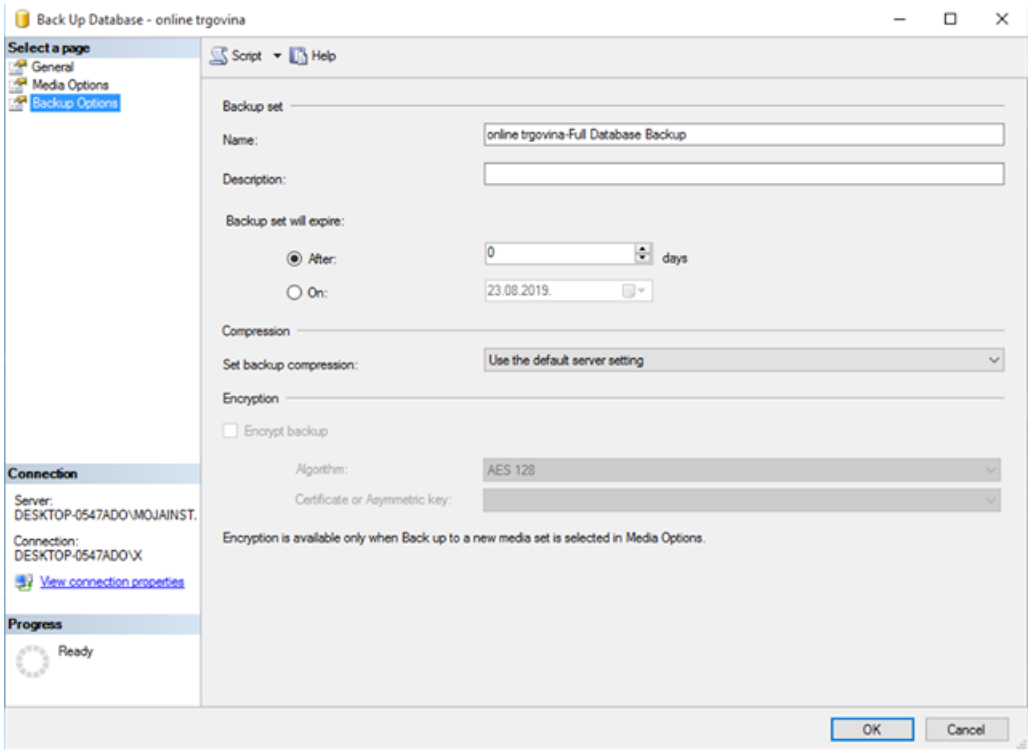
Slika 3.37: Postavke rezervne kopije

U stavci ”Media Options” koja je prikazana na slici 3.38 definiramo želimo li novu rezervnu kopiju dodati na kraj datoteke rezervne kopije ili tu datoteku želimo prepisati novom rezervnom kopijom. Tako bi se u prvom slučaju unutar jedne datoteke mogle nalaziti višestruke rezervne kopije (bilo kojeg tipa), a u drugom slučaju tek zadnja rezervna kopija. Rezervnu kopiju je uvijek poželjno provjeriti nakon kreiranja kako bi se sa sigurnošću znalo je li čitljiva i može li se koristiti za povrat baze podataka, a to se može postići odabirom stavki ”Verify backup when finished” i ”Perform checksum before writting to media”. Iz istog razloga najčešće nije poželjno nastaviti s izradom rezervne kopije u slučaju greške, tj. odabirom stavke ”Continue on error”.



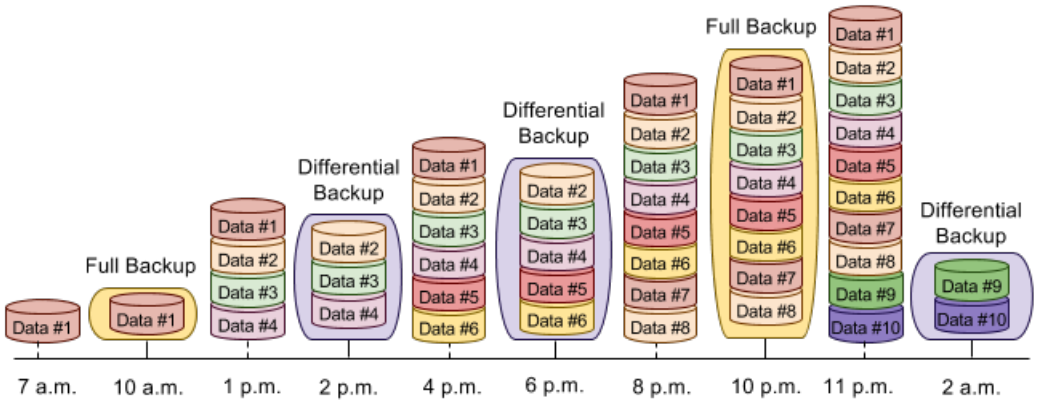
Slika 3.38: Postavke rezervne kopije

Odabirom stavke ”Backup Options” otvara se obrazac koji je prikazana na slici 3.39. Tu se može odabrati i vijek trajanja, a u tom razdoblju (definiranom brojem dana ili krajnjim datumom) datoteka rezervne kopije ne može biti prepisana drugom rezervnom kopijom. Moguće je koristiti i kompresiju datoteka rezervne kopije kako bi se smanjila njihova veličina.



Slika 3.39: Postavke rezervne kopije

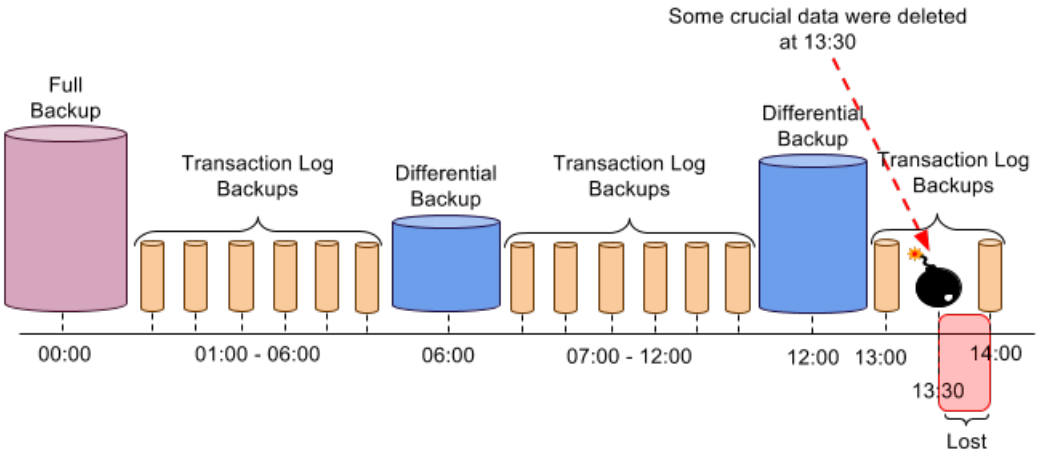
Potupune rezervne kopije zbog svoje veličine često nisu najbolje rješenje kada je riječ o velikim bazama podataka. Zbog uštede prostora na disku koriste se diferencijalne rezervne kopije koje sadrže samo izmjene koje su nastale nakon zadnje potpune rezervne kopije. Svaka diferencijalna rezervna kopija će u sebi sadržavati i sadržaj svake prethodne diferencijalne rezervne kopije, i tako sve dok se ne napravi sljedeća potpuna rezervna kopija što možemo vidjeti na slici 3.40. Bez obzira na postojanost diferencijalne rezervne kopije za povrat baze podataka i dalje je potrebna potpuna rezervna kopija koja se prva vraća, a tek nakon nje odgovarajuća diferencijalna rezervna kopija.



Slika 3.40: Potpune i diferencijalne rezervne kopije

Diferencijalna rezervna kopija se kao i potpuna rezervna kopija može izgraditi korištenjem SSMS alata ili izvršavanjem SQL upita.

U kombinaciji s potpunim i diferencijalnim rezervnim kopijama najčešće se koriste i rezervne kopije dnevnika transakcija (*transaction log backups*). Rezervna kopija dnevnika transakcija sadrži sve zapise dnevnika koji nisu bili uključeni u posljednju sigurnosnu kopiju dnevnika transakcija, za razliku od diferencijalnih koje će u sebi sadržavati i sadržaj svake prethodne diferencijalne rezervne kopije. Moguće ih je izrađivati samo ukoliko baza podataka prethodno ima izrađenu potpunu rezervnu kopiju te ukoliko koristi potpuni model oporavka. Kad je izrađena potpuna rezervna kopija onda ih je moguće izrađivati i iza diferencijalnih rezervnih kopija. Također, jedino ovaj tip rezervne kopije omogućuju povrat baze podataka u željenu točku u vremenu, što je prikazano na slici 3.41. Kao i prethodne možemo ih izgraditi korištenjem SSMS alata ili izvršavanjem SQL upita.

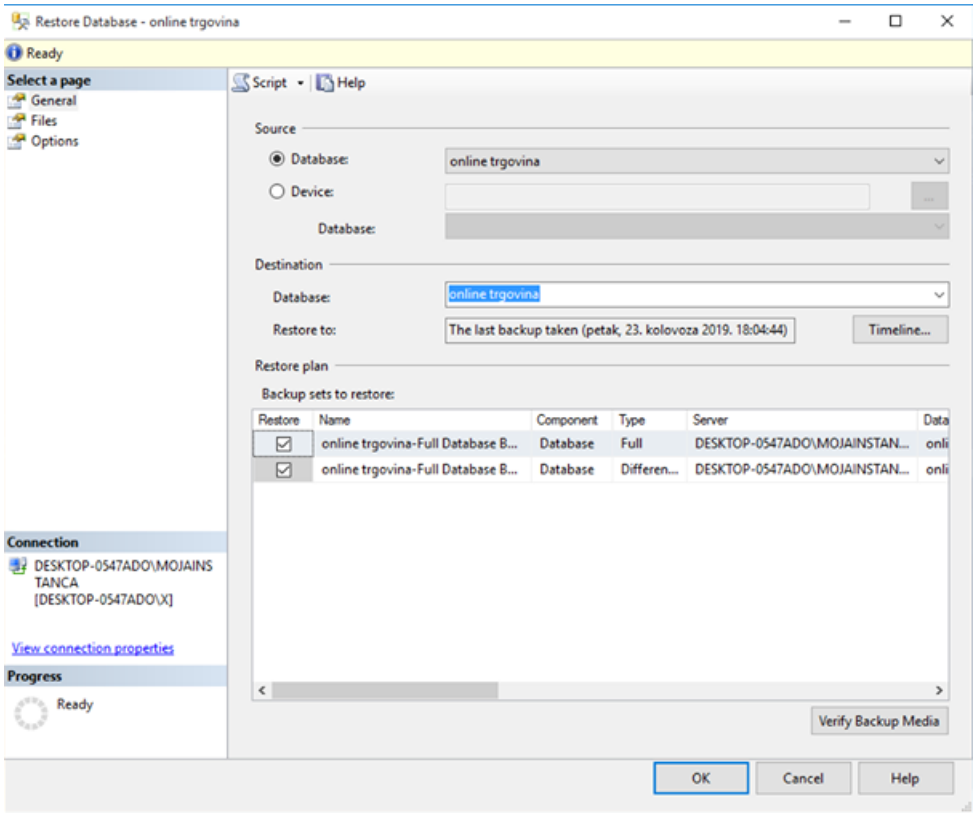


Slika 3.41: Povrat baze u željenu točku u vremenu

Ovisno o veličini baze podataka, broju transakcija u zadanom vremenu, osjetljivosti podataka i sigurnosnoj politici tvrtke može postojati veliki broj različitih strategija izrade rezervnih kopija.

3.5.2 Povrat baze podataka

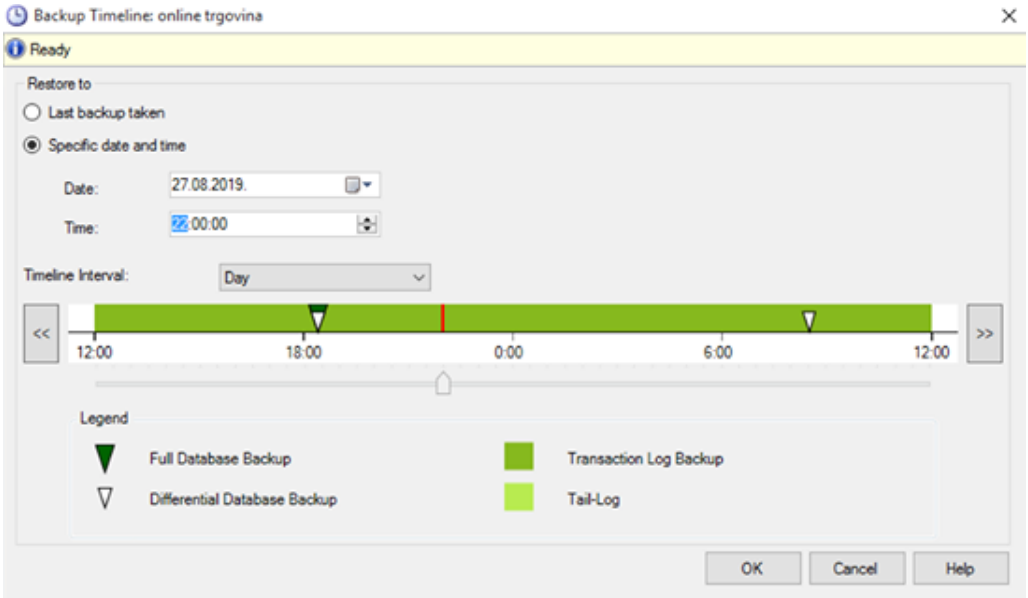
Prilikom povrata baze podataka treba vratiti kompletan lanac rezervnih kopija koji vodi do željenog datuma i vremena. Na primjer, za povrat baze podataka u najnovije vrijeme prvo je potrebno vratiti zadnju potpunu rezervnu kopiju, a nakon nje i zadnju diferencijalnu rezervnu kopiju. Tek nakon toga treba vratiti sve rezervne kopije dnevnika transakcije koje su iz vremena nakon zadnje diferencijalne rezervne kopije. Na slici 3.42 prikazujemo povrat baze podataka u najnovije vrijeme.



Slika 3.42: Povrat baze podataka

Za svaku rezervnu kopiju moguće je vidjeti odakle počinje i završava, odnosno informacije o početnim i završnim log sekvencama (*log sequence numbers*, LSN). Također je moguće vidjeti i vezu između potpune i diferencijalnih rezervnih kopija uspoređujući potpune LSN brojeve (*full LSN*) i LSN brojeve kontrolnih točki (*checkpoint LSN*).

Za povrat baze podataka u točno određenu točku u vremenu (datum i vrijeme) potrebno je imati odgovarajuće rezervne kopije dnevnika transakcija. Zatim, klikom na gumb ”Timeline” pojavljuje se sljedeći obrazac koji je prikazana na slici 3.43.



Slika 3.43: Povrat baze podataka u točno određenu točku vremena

Na grafički način prikazane su sve već postojeće rezervne kopije, a klikom na gumb ”OK” lanac rezervnih kopija automatski će se reorganizirati tako da se omogući povrat baze podataka u odabranu točku u vremenu,a to uključuje automatski odabir odgovarajuće potpune, diferencijalne i svih potrebnih rezervnih kopija dnevnika transakcija.

Jedna od najvažnijih stavki povrata baze podataka je status oporavka, a on može biti jednog od ova tri tipa:

- *RESTORE WITH RECOVERY*. Baza podataka je odmah nakon završenog povrata rezervne kopije dostupna korisnicima.
- *RESTORE WITH NORECOVERY*. Koristi se za vraćanje višestrukih rezervnih kopija. Korisnici nemaju pristup bazi podataka sve dok se ne povrate sve željene rezervne kopije. Nakon povrata zadnje rezervne kopije bazu podataka potrebno je postaviti u oporavljeno stanje naredbom RESTORE.
- *RESTORE WITH STANDBY*. Baza podataka dostupna je samo za čitanje. Sve nedovršene transakcije neće biti će izvršene, ali će biti spremljene u zasebnu datoteku za slučaj da se možemo vratiti u vrijeme prije zadnjeg povrata baze podataka.

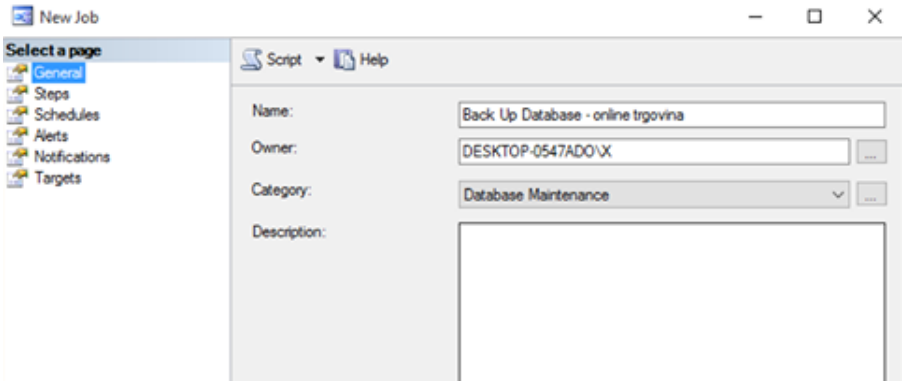
3.5.3 SQL Server Agent

Uz odgovarajuću strategiju izrade rezervnih kopija održavanje baze podataka podrazumijeva izradu rezervnih kopija, provjeru integriteta, čišćenje nepotrebnih datoteka i tako dalje. Da bi se u SQL Serveru automatizirao proces redovnog održavanja baza podataka moguće je koristiti komponente SQL Server Agent servisa zajedno s planovima održavanja. SQL Server Agent je Windows servis zadužen za automatizaciju administrativnih poslova u SQL Serveru. Postoji kao sastavni dio svake instance i on se ne pokreće automatski podizanjem operacijskog sustava. Međutim, te postavke moguće je naknadno izmijeniti upotrebom SQL Server Configuration Manager aplikacije, u popisu svih Windows servisa ili ga definirati prilikom instalacije instance.

SQL Server Agent sastoji se od četiri komponente: poslovi (*jobs*), rasporedi (*schedules*) upozorenja (*alerts*) i operateri (*operators*).

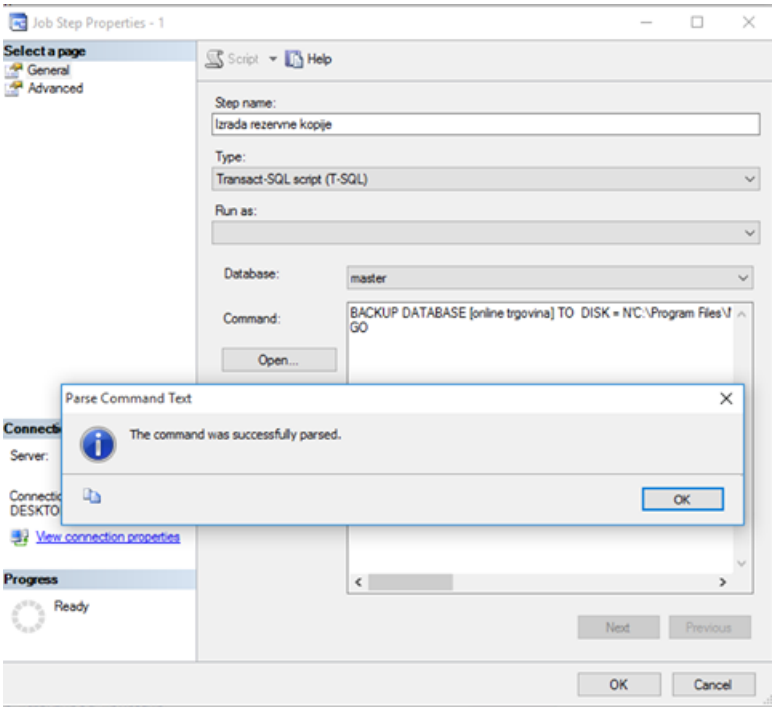
Poslovi su najvažnija komponenta SQL Server Agenta jer se pomoću njih definiraju administrativne zadaće. Svaki posao izvršava se na osnovu rasporeda, generiranog upozorenja ili na zahtjev (izvršavanjem uskladištene procedure sp_start_job). Posao može biti i jednokratan i on se automatski briše nakon uspješnog završetka.

Novi posao se može kreirati desnim klikom na "Jobs" i odabirom stavke "New Job". Klikom na stavku "New Job" prikazuje se obrazac prikazan na slici 3.44, gdje vidimo da se prilikom kreiranja posla može definirati ime, vlasnik, kategorija i opis.



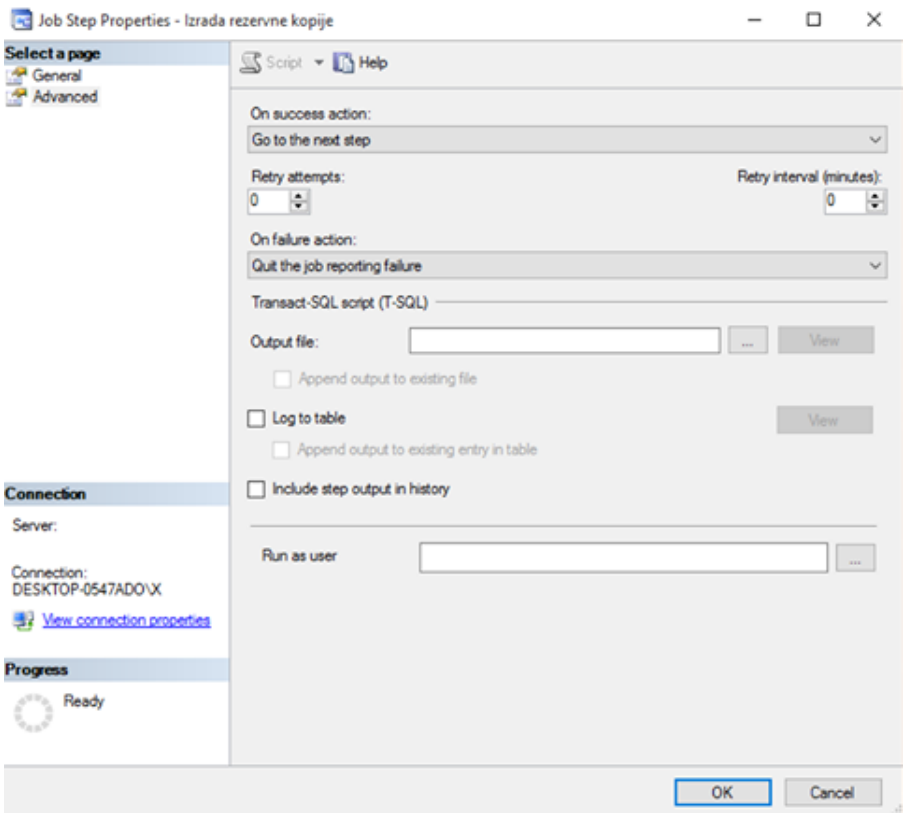
Slika 3.44: Kreiranje posla u SQL Server Agentu

Svaki posao se sastoji od barem jednog koraka, a izrada koraka prikazana je na slici 3.45. U primjeru sa slike cilj našeg posla je napraviti potpunu rezervnu kopiju online trgovine baze podataka, a za to je potrebno napraviti korak tipa "Transact-SQL script (T-SQL)" te u polje "Command" kopirati odgovarajući T-SQL kod za izradu potpune rezervne kopije. Ispravnost T-SQL koda moguće je provjeriti klikom na gumb "Parse" nakon čega se dobije odgovarajuća povratna informacija koja je prikazana na slici 3.45.



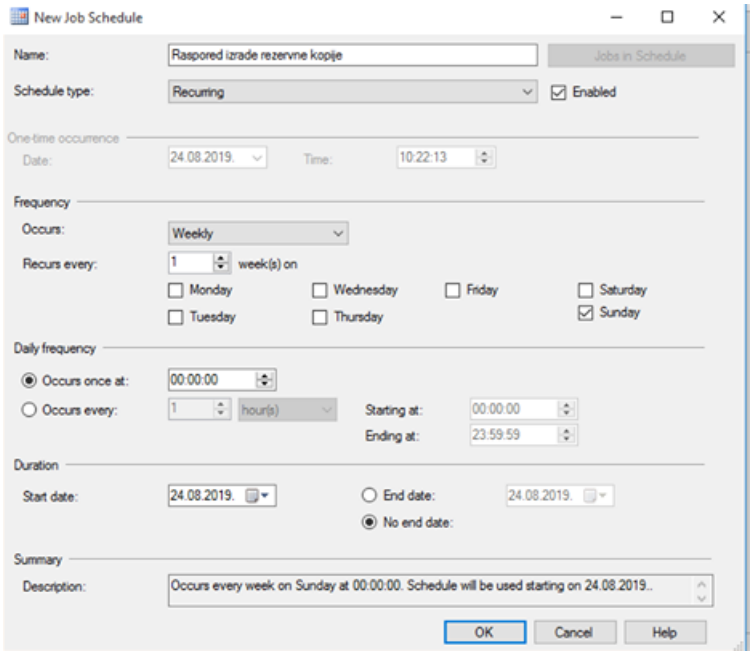
Slika 3.45: Kreiranje koraka posla

U kartici ”Advanced” prikazanoj na slici 3.46 možemo definirati akcije nakon uspješnog ili neuspješnog završetka koraka. Postoji i mogućnost ponovnog izvršavanja koraka.



Slika 3.46: Postavke koraka posla

Posao izrade rezervnih kopija je poželjno izvršavati periodički, odnosno unaprijed definiranim rasporedom. Na slici 3.47 je prikazano kreiranje rasporeda izvršavanja posla, s tim da se svaki posao može izvršavati sukladno jednom ili više rasporeda. Tipom rasporeda definira se trenutak i učestalost izvršavanja posla. Može se raditi o poslu koji se ponavlja, izvršava samo jedanput ili se izvršava u posebnim situacijama. Izvršavanje posla može biti na dnevnoj, tjednoj ili mjesečnoj bazi, sa ili bez krajnjeg datuma izvršavanja.



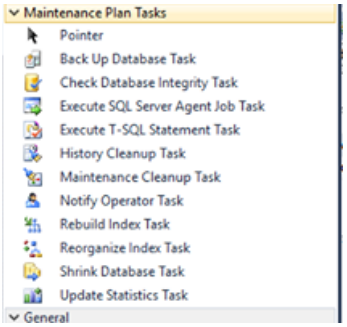
Slika 3.47: Kreiranje rasporeda izvršavanja posla

Posao može biti izvršen i kao rezultat generiranog upozorenja. Upozorenje je automatski odgovor na određeni događaj, pri čemu može biti riječ o SQL Server događaju, problematičnim performansama izvedbe ili WMI (Windows Management Instrumentation) događajima. Upozorenje se može generirati prilikom neuspješno obavljenog posla, nedostatka resursa, kvara na sklopovlju i tako dalje, a u tom trenutku se može kontaktirati i operater. Operater je jedna od osoba koja je zadužena za administriranje i održavanje SQL Servera.

3.5.4 Plan održavanja

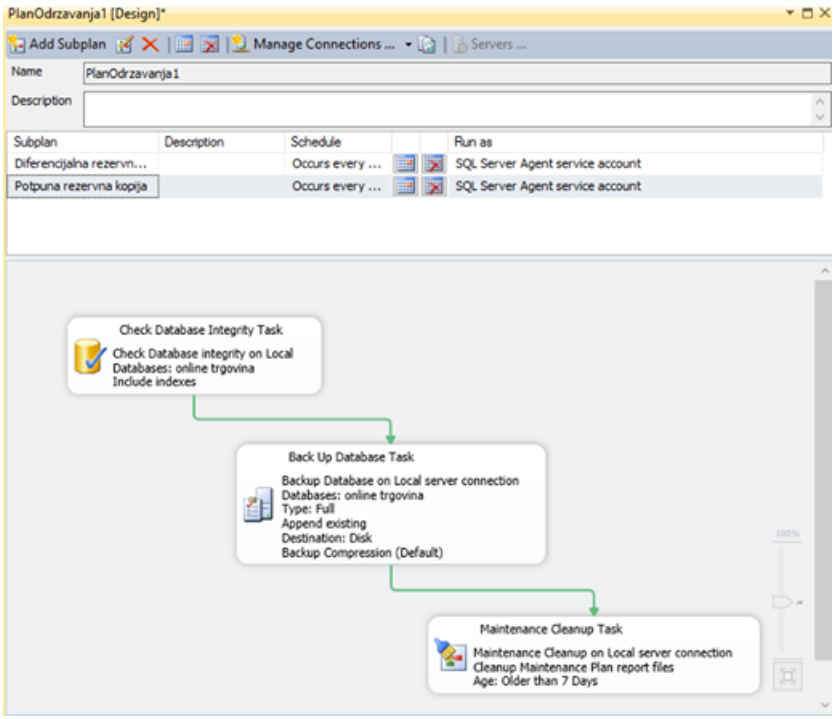
Plan održavanja (*maintenance plan*) na vizualan način omogućuje implementaciju, pregled i izmjenu strategije održavanja. Koraci unutar plana održavanja su vizualno prikazani tokom iz kojeg su jasno vidljivi strategija i scenariji održavanja. Moguće ga je kreirati korištenjem SSMS alata upotrebom stavke ”Management / Maintenance Plans / New Maintenance Plan...”.

Prilikom izrade plana održavanja moguće je koristiti 11 unaprijed definiranih tipova zadataka koji su prikazani na slici 3.48. Ovisno o strategiji održavanja, zadatke je moguće proizvoljno kombinirati. Pomoću toka se definira njihov redoslijed izvršavanja, a on može biti i uvjetno definiran, uzimajući u obzir da se neki od zadataka možda neće uspješno izvršiti.



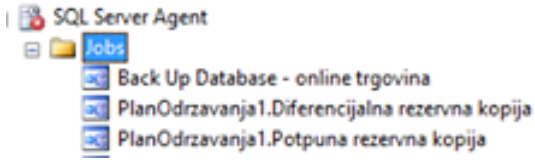
Slika 3.48: Popis zadataka plana održavanja

Svaki plan održavanja može imati jedan ili više podplanova. U primjeru sa slike 3.49 plan održavanja online trgovine baze podataka sastoji od dva podplana, svakog za pojedini tip rezervne kopije. Podplan za izradu potpune rezervne kopije počinje provjerom integriteta baze podataka. U slučaju uspjeha tog zadatka prelazi na izradu rezervne kopije. Povijest izvršavanja pojedinih planova i podplanova moguće je pregledati u SSMS alatu odabirom stavke "Management / Maintenance Plans / View History".



Slika 3.49: Izrada plana održavanja baze podataka online trgovina

Za svaki podplan kreira se SSIS (SQL Server Integration Services) paket koji se izvršava SQL Server Agent poslom, bilo automatski ili na zahtjev. Zbog plana održavanja koji je definiran slikom biti će kreirana dva zasebna SQL Server Agent posla. Ti poslovi su prikazani na slici 3.50.



Slika 3.50: SQL Server Agent poslovi održavanja

3.5.5 Kopiranje baze podataka

Za veliki broj situacija javlja se potreba za kopiranjem. Na primjer, prilikom razvoja aplikacija, nadogradnje baze podataka ili testiranja.

Imamo četiri metode kopiranja baze podataka, a to su:

- Metoda ”odspoji-kopiraj-spoji”
- Skriptiranje baze podataka
- Čarobnjak za kopiranje baze podataka
- Izrada i povrat rezervne kopije

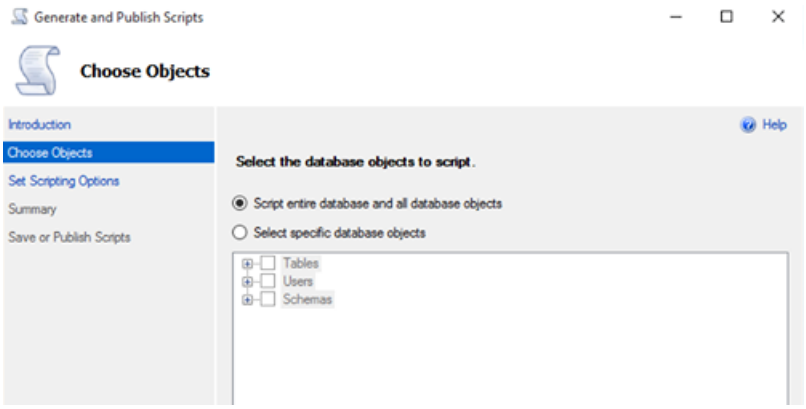
Odabir odgovarajuće metode ovisi o veličini baze podataka, potrebi za dostupnošću tijekom kopiranja, sadržaju, strukturi i brzini kopiranja.

Metoda ”odspoji-kopiraj-spoji” koristi se kada se datoteka baze podataka fizički kopira s jedne lokacije na drugu. Da bismo na takav način mogli kopirati bazu podataka prvo ju trebamo odspojiti od instance SQL Servera (desni klik na bazu podataka, stavka ”Tasks / Detach...”). Nakon kopiranja baze podataka potrebno ju je ponovno prispojiti instanci (desni klik na popis baza podataka, stavka ”Attach...”). Nedostatak ove metode je nedostupnost baze podataka tijekom kopiranja, odnosno sve dok ju se opet ne pripoji instanci SQL Servera. Zbog toga se ova metoda koristi kada je broj upita prema bazi podataka minimalan ili ih uopće nema.

Ponekad je potrebno kopirati samo strukturu baze podataka, tj. definicije tablica, pogleda, uloga i tako dalje, a to se može postići pomoću SSMS alata generiranjem dgovarajuće SQL skripte na način da desnim klikom kliknemo na bazu podataka i upotrebimo

stavke ”Tasks / Generate Scripts. . . ”. Izvršavanjem generirane skripte u drugom (testnom) okruženju kreirat će se navedena baza podataka i svi njeni odabrani objekti.

Sa slike 3.51 vidimo da je SQL skriptu moguće generirati za kompletnu bazu podataka ili samo za neke od njenih objekata. Generiranu SQL skriptu moguće je iz SSMS alata objaviti na web servisu, spremiti u jednu ili više datoteka, spremiti u međuspremnik ili prikazati u novom uređivačkom prozoru.



Slika 3.51: Izrada skripte za kompletnu bazu podataka

Prilikom odabira odredišta skripte moguće je kliknuti i na gumb ”Advanced” gdje se mogu urediti neke od naprednih postavki za generiranja skripte. Postavkom ”Types of data to script” moguće je generirati skriptu samo za strukturu baze podataka odabirom stavke *shema only*, samo za podatke sadržane u objektima odabirom stavke *data only* ili za oboje odabirom stavke *schema and data*. U skriptu je moguće uključiti i definicije prijavnih naloga, okidača, statistika itd.

Generiranje SQL skripte ne zahtjeva odspajanje baze podataka te je uvijek prikladno za kopiranje njene strukture, ali generiranje SQL skripte u svrhu kopiranja podataka ipak nije preporučljivo jer traje iznimno duže nego bilo koja druga metoda kopiranja sadržaja baze podataka.

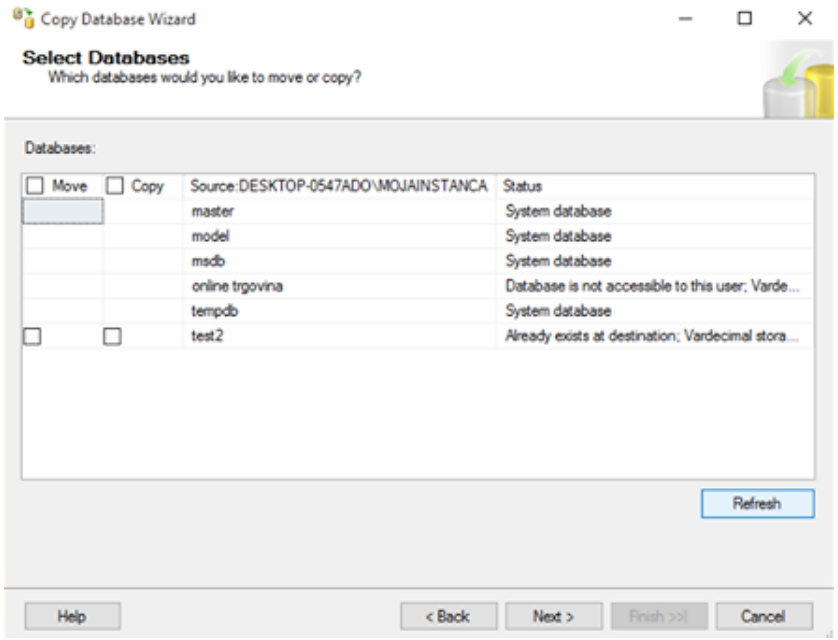
Treća metoda je čarobnjak za kopiranje baze podataka, a pokreće ga se desnim klikom na bazu podataka te odabirom stavke ”Tasks / Copy Database. . . ”. Nakon toga potrebno je odabrati metodu prijenosa podataka.

Prvom metodom (”Use the detach and attach method”) radi se identičan postupak kao i metodom ”odspoji-kopiraj-spoji”. Baza podataka se prvo odspoji od instance SQL Servera, a nakon toga se njene datoteke fizički kopiraju na lokaciju druge instance. Obje baze podataka zatim se automatski pripoje svojim instancama.

Druga metoda ”Use the SQL Management Object method” podsjeća na kreiranje SQL skripte. Prvo se čita definicija svakog objekta iz izvorišne baze podataka, a nakon toga se

na osnovu definicije takav objekt kreira u odredišnoj bazi podataka. Na kraju se kopiraju podaci i ponovno se kreiraju indeksi, metapodaci itd.

U sljedećem koraku koji je prikazan na slici 3.52 moguće je odabrati jednu ili više baza podataka iz izvorišne instance i za svaku od njih odabrati operaciju kopiranja (*copy*) ili prijenosa (*move*) u odredišnu instancu. Ukoliko odredišna baza podataka već postoji u drugoj instanci, čarobnjak će ponuditi mogućnost promjene imena nove odredišne baze podataka.



Slika 3.52: Odabir baze podataka

Unatoč mnogim automatiziranim značajkama ovog čarobnjaka, vrlo često postupak kopiranja nije uspješno izvršen. Da bi se to izbjeglo potrebno je osigurati da je SQL Server Agent pokrenut u odredišnoj instanci, da su podatkovne datoteke i datoteke dnevnika transakcija izvorišne baze podataka dostupne iz instance odredišne baze podataka itd.

Metoda izrada i povrat rezervnih kopija je jedna od vrlo čestih metoda kopiranja baze podataka. Takva (potpuna) rezervna kopija najčešće se kreira na način da ne utječe na već postojeći lanac rezervnih kopija izvorišne baze podataka (*Copy-only Backup*), a u drugoj instanci njen povrat moguće je napraviti pod istim ili drukčijim imenom.

Bibliografija

[1] C.S. Mullins, *Database Administration- The Complete Guid to DBA Practices and Procedures*. Second Edition. Addison-Wesley, Upper Saddle River NJ, USA, 2012.

[2] P. Berzukov, *Understanding database Administration*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 2010.

[3] A. Jorgensen, B. Ball, S. Wort, R. LoForte, B. Knight, *Professional Microsoft SQL Server 2014 Administration*. John Wiley and Sons, Indianapolis IN, USA, 2014.

[4] R. Manger, *Baze podataka*. Element, Zagreb, 2012.

[5] Visual Paradigm. Online version. https://online.visual-paradigm.com/

[6] Microsoft SQL Server 2014. Version ”Express”. https://www.microsoft.com/

[7] Microsoft. HASHBYTES (Transact-SQL). https://docs.microsoft.com/en-us/sql/t-sql/functions/hashbytes-transact-sql?view=sql-server-2017

Sažetak

Tema ovog diplomskog rada je održavanje baze podataka. Diplomski rad je podijeljen u tri poglavlja.

U prvom poglavlju su opisani poslovi koje obavlja administrator baze podataka, a to su: oblikovanje baze podataka, praćenje i podešavanje performansi, osiguravanje dostupnosti baze korisnicima, čuvanje sigurnosti podataka, stvaranje rezervnih kopija baze podataka, oporavak baze u slučaju njezinog oštećenja i osiguravanje integriteta podataka.

U drugom poglavlju se daje pregled kategorija i vrsta proizvoda dostupnih DBA-u kao pomoć u upravljanju i održavanju baza podataka.

U trećem poglavlju je prikazan studijski primjer baze bodataka, koji se sastoji od izrada baze podataka i njezinih objekata. Prikazani su načini na koji se baza organizira, te kako se održava i nadzire baza podataka. Također je prikazano kako se osigurava sigurnost. Za izradu praktičnog rada koristili smo Visual Paradigm i Microsoft SQL Server.

Summary

The theme of this thesis is database administration. The thesis is divided into three chapters.

The first chapter describes the tasks that the database administrator performs, and those are: database design, monitoring and tuning of performance, ensuring database availability to users, maintaining security, backing up the database, recovering the database in the event of damage, and ensuring data integrity.

Chapter two provides an overview of the categories and types of products available to the DBA to help manage and maintain databases.

Chapter three presents a case study of a database, which consists of creating a database and its objects. The ways are presented in which the database is organized and how the database is maintained and monitored. Database security is also shown. We used Visual Paradigm and Microsoft SQL Server to create the practical work.

Životopis

Rođena sam u Posušju 26. sječnja 1996. godine. Osnovnu i srednju školu sam pohađala u Posušju, gdje sam završila i osnovnu glazbenu školu. Nakon završene srednje škole upisala sam 2014. godine preddiplomski sveučilišni studij Matematike na Prirodoslovno-Matematičkom fakultetu u Zagrebu. Nakon završenog preddiplomskog studija sam 2017. godine upisala Diplomski sveučilišni studij Računarstvo i matematika na istom fakultetu.