Факултет инжењерских наука Универзитета у Крагујевцу

Анђела Д. Мијаиловић

Архивирање података

Дипломски рад

Крагујевац, 2020.

Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство
Ниво студија: Основне академске студије
Предмет: Базе података
Број индекса: 561/2016

Анђела Д. Мијаиловић

Архивирање података

Дипломски рад

Комисија за преглед и одбрану:

1. **Проф. др Милан Ерић - ментор**
2. _____ Датум одбране:_____
3. _____ Оцена:_____

Изјава о самосталној изради рада

Изјављујем да сам овај дипломски рад урадила самостално, служећи се стеченим знањем и искуством као и наведеном литературом.

561/2016 Мијаиловић Анђела

_____

(потпис)

Универзитет у Крагујевцу
Факултет инжењерских наука

Основне студије: **Рачунарска техника и софтверско инжењерство**

Име и презиме: **Анђела Мијаиловић**

Број индекса: 561/2016

ПРИЈАВА ДИПЛОМСКОГ РАДА

Тема рада: ***Архивирање података***

Задатак: У оквиру дипломског рада обухватити: Уводна разматрања у вези архивирања података (поузданост, ефикасност, доступност, брзина приступа ...) ; Архивирање података као изазов; Традиционалне приступе архивирања; Сервисни приступ; Конкретан студијски пример примене; Закључна разматрања.

Ментор:

_____

Крагујевац, 13.03.2020. Проф. др Милан Д. Ерић

Садржај

Abstract

This paper provides a general overview of data archives through time, as well as security of databases especially backing up and restoring. Paper also describes different types , models, methods and strategies of backups. Example of database backup and restore is given in this paper. Programs that were used for database example is SQL Server.

Key words: backup, security, databases

Резиме

У овом раду је дат општи опис архива података кроз историју, као и безбедност савремених база података са посебним освртом на прављење резервних копија и њихова употреба. Описане су различите врсте модела резервних копија података, метода као и стратегија за планирање процеса архивирања. Програм, коришћен у овом раду, у коме је описан конкретан пример прављења резервне копије базе, као и њено коришћење је SQL Server.

Кључне речи: архивирање података, безбедност, база податак

1. Увод

Компјутери и технологије су постали саставни део друштва, а њихова употреба свакодневница. У модерним државама, целокупна инфраструктура је аутоматизована до одређеног степена чинећи да се свакодневне животне активности одвијају брже и лакше. Управо се у таквом начину живота крије и највећа рањивост. Престанак протока информација на критичној инфраструктури било да је реч о природној катастрофи, квару или нападу који је произвео појединац или група, може државу доста уназадитити и изазвати велике проблеме и губитке. Управо због тога се у доктринарним документима водећих војних и економских сила, акценат ставља на заштиту најважнијих инфраструктурих сегмената, а посебно на информацију и комуникацију.

Исто као што држава штити информатичку инфраструктуру, тако и компаније стављају акценат на заштиту информација. Савремено пословање је незамисливо без комуникација и база података. Базе података представљају извор информација за већину апликација којима помажу у обављању основних активности. Ширењем тржишта, а самим тим и развојем организација, без обзира да ли су оне државне или приватне, уочена је примена система база података као кључне технологије за управљање подацима и помоћног средства за доношење одлука.

Зато ће се овај рад бавити итраживањем базе података са посебним освртом на ахивирање података односно прављење резервних копија и њихово коришћење. На конкретном примеру је приказано прављење резервне копије базе података кроз програм SQL Server.

1. Основни појмови и дефиниције

1.1 Податак и информација

У свакодневном разговору, не правимо увек јасну разлику између термина податак и информација. Међутим, разлика између ова два појма је од виталног значаја за разумевање појма информациони систем.

Податак је низ чињеница, које су без значаја ако се посматрају засебно. Подаци могу бити структуирани и неструктуитани односно мултимедијални. Структуирани подаци су бројеви, карактери и датуми (време). Поред структуираних података постоје и друге врсте података као што су разна документа, мапе, фотографије,па чак и видео записи.[8]

У свакодневном животу се срећемо са различитим проблемима. За решавање проблема некад постоји само једно решење, а некад их има више. Кад има више решења онда морамо донети одлуку које ћемо решење применити. Одлуку доносимо на основу познатих информација везаних за дати проблем. Информације се добијају из већег броја чињеница односно података. Може се рећи да су информације подаци представљени у облику који је погодан за доношење одлука.

Према томе, **информација** је скуп чињеница тако обрађених и организованих да представљају неко обавештење. **Обрада података** се може посматрати као скуп активности којима се подаци трансформишу у информације. Можемо још рећи да су подаци сировине из које се обрадом добијају информације.

Циљ информационог система је да обради колекцију низа чињеница на неки начин, тако да продукује информацију која може да користи неком.

Пример. Ако би се телефонски именик састојао од случајног избора имена, адреса и телефонских бројева, без одређеног редоследа, и без логичке повезаности између имена, адреса и бројева телефона, био би потпуно без користи. Без обзира што садржи све податке, такав именик је „информационо“ безвредан. Међутим, ако повежемо сваки телефонски број са именом и адресом одговарајућег претплатника, добили смо информацију. Ако још алфабетски уредимо информације о претплатницима, обезбедили смо врло ефикасан приступ свакој информацији.

Овај пример је једноставна илустрација комплексности једног информационог система. [8]

1.2 Информациони систем

Студент је део едукационог система, без обзира на то шта студира и колико је успешан у томе. С друге стране, само људско тело је један врло комплексан биохемијски систем, као и досадни комарац у соби или биљка која мирно стоји у саксији поред прозора.

Пошто је систем субјективни концепт, не постоји опште прихваћена дефиниција система. У циљу ближег проучавања система прихватићемо следећу дефиницију.

- Систем је организовани скуп компоненти са посебним везама између њих.
- Систем ради нешто, тј. представља модел понашања, јединствен за тај систем, или има специфичну сврху или циљ.
- Свака компонента утиче на одређени начин на понашање система као и на његово постојање. Ако се нека компонента уклони, промениће се и понашање система.

Не чини свака случајна колекција елемената систем, као што ни случајна колекција речи не чини смислену реченицу. Потребно је да постоји одређена структура која има неки облик уређења, модел и сврху.

Информациони систем је систем који прикупља и обрађује податке са циљем да продукује информације за своје крајње кориснике. Да би информациони систем функционисао успешно, његови аутори и корисници се морају сложити у следећем: шта је сврха система, које су његове компоненте и какве су везе међу компонентама.[8][4]

1.3 База података

Појам база података појавио се крајем шездесетих година двадесетог века. Овај појам је означавао скуп међусобно повезаних података који се чувају заједно, и међу којима има само онолико понављања колико је неопходно за њихово оптимално коришћење.
Подаци се чувају тако да буду независни од програма који их користе, и структуирају се тако да је омогућен пораст базе.
После сваке активности над базом, стање базе мора бити конзистентно (ваљано). Дакле, подаци у бази и односи међу њима морају задовољавати унапред задате услове који осликавају део реалности моделиране у бази.
Обрада података није везана за програмски језик опште намене, већ за виши „упитни“ језик.
Базе података представљају скуп повезаних и структуираних података, која постоји релативно дуго и коју користи и одржава више корисника, односно програма (апликација).[2]

1.4 Систем за управљање базом података

Систем за управљање базама података(СУБП) представља софтверски систем за чување и претраживање података. Налази се између корисника и физичких података у бази. Он омогућава ефикасно формирање, коришћење и мењање базе података.[2][8]
Неке од улога које обезбеђује СУБП су:

- Складиштење података са минималном редундансом
- Поузадност података у случају отказа
- Поуздано паралелно коришћење података од стране овлашћених лица
- Логичка и физичка независност података
- Једноставно комуницирање са базом путем упита

СУБП је обично заснован на неком теоријском моделу и подржава програмске језике за:

- дефинисање структуре и услова интегритета базе података (DDL, Data Definition Language)
- селекцију и измене садржаја базе података (DML, Data Modification Language)

Предности рада са базом података:

- Интегрисаност података – централизована контрола свих података и управљање подацима на систематизован начин,
- Независност података од програма који их обрађују,
- Раздвајање физичког записа и логичке организације података.
    - Поузданост података – остварује се очувањем интегритета података и контролом приступа подацима.
    - Интегритет података – тачност или коректност података.

1.5 Циљеви управљања подацима

- Расположивост података
    - Администратор базе података је задужен да уради све како би обезбедио да „систем увек буде расположив“.
    - Расположивост се практично дефинише као одзивност система у неким гарантованим границама (у овом контексту се разматра само време одзива).
    - Систем обично није расположив управо када је потребан. У време када је расположивост најпотребнија, онда је и најтеже остварива, зато што је тада систем највише оптерећен.[3]
- Интегритет података
    - Појам интегритет се у контексту база података односи на прецизност, пуноважност и коректност података у бази.
    - Одржавање интегритета података је од велике важности за СУБП.
    - Уколико један или више корисника у исто време покушава да приступи и промени податке у једној табели, може доћи до преплитања акција и могућности неконзистентне промене или нетачности података. У циљу решавања овог проблема, СУБП подржава коришћење транскација како би се обезбедило конкурентно извођење акција као и опоравак базе података у случају грешке.
    - Трансакција је низ операција над базом података који одговара једној „логичкој јединици посла“. Основна својства трансакције су атомичност, трајност, изолованост, могућност испреплетаног извршавања. Свака трансакција мора да штити конзистентност базе података.
    - Уколико више корисника истовремено (конкурентно) захтева приступ истим подацима, ови захтеви могу бити конфликтни па их СУБП мора разрешити како би спречио нарушавање интегритета базе.
    - Ако је база података конзистентна, као и трансакција. База ће остати конзистентна и након извршења трансакције.
    - Конзистентност трансакције је одговорност корисника, а конзистентност базе након завршетка конзистентне трансакције је одговорност СУБП.[3]
- Заштита података
    - Како би се база података безбедно и ефикасно користила, сваки корисник мора имати ауторизацију за приступ СУБП, односно, сваком кориснику мора да буде додељено корисничко име којим се представља систему. Поред овог првог нивоа сигурности базе, СУБП обично подржава и више нивое безбедности.
    - Тако се корисници или групе корисника могу разликовати према нивоу ауторизације, односно, може им се омогућити да обављају различите опште задатке над базама података, као што су повезивање са базом, креирање табела или администрирање система.
    - Различитим корисницима или групама корисника се могу доделити различита права или привилегије приступа и извршавања специфичних операција над специфичним објектима у бази. На пример, да би неки корисник могао да мења податке у некој табели, он мора да има право (експлицитно или имплицитно додељено) за промену података у тој табели.[3]
- Независност од апликација
    - скривеност физичке структуре - ако дође до промена у физичкој структури базе, то неће захтевати промене у апликацијама
    - скривеност логичке структуре - ако дође до промена у логичкој структури базе, то неће захтевати промене у апликацијама[3]

2. Традиционални и сервисни приступ архивирању

Када су рачунари почели да се користе за обраду података, нису постојале базе података. Рачунари су у то време били знатно слабији него данашњи, заузимали су читаву просторију и користили су се искључиво за научна истраживања.
Постепено су рачунари почели да се уводе у пословни свет. Да би били од користи за пословне апликације, рачунари су морали да складиште, манипулишу и преузимају велике датотеке података. Како су пословне апликације постајале све комплексније, постало је очигледно да класични системи засновани на датотекама имају великих недостатака и ограничења. У већини битних пословних апликација данас се уместо традиционалног (класичног) система заснованог на датотекама користи база података.

У традиционалном приступу са датотекама, програми су директно повезани са датотекама, сваки програм мора да познаје детаљан запис података на диску.

Као функционисања овог система, посматрајмо информациони систем једног универзитета. Стандардна функционалност таквог система је да студенти преко интернета пријављују испите, професори након испита уносе оцене. У класичној обради података са датотекама постојао би велики број датотека за студенте и за професоре. Неке од датотека би биле заједничке.
Апликације за пријаву испита и унос оцена би биле оптерећене великим бројем прецизнх дефиниција датотека. Ове дефиниције(формат и начин приступа датотеци) би неминовно проширивале основни код апликације, тако да би сама функционалност око пријаве испита и уноса оцена била практично у другом плану. Како се у некој апликацији повећава број линија кода, сама апликација постаје непрегледна и склона грешкама, од којих су најтеже тзв. логичке грешке. Тестирање таквих апликација и исправљање логичких грешака је веома сложено. Пошто посоји велики број датотека, неминовно постоји и велика редуданса(понављање) података. Исти подаци се могу наћи у различитим датотекама, а одржавање таквих података је тешко и грешке у раду су неминовне.[3]

2.1 Недостаци система заснованог на датотекама

Постоји више мана које су типичне за овај систем. На првом месту то је **зависност између програма и података**. Описи датотека се чувају у оквиру сваког програма који приступа тој датотеци. Као последица овога, свака промена која се направи у датотеци, а односи се на структуру, моментално подразумева да се мора мењати и опис датотека у сваком програму који приступа тим подацима.

Претпоставимо да се величина поља „Адреса студента“ мења са 20 карактера на 30 карактера. Због тога се опис у сваком програму мора ажурирати. Често је тешко и само лоцирање свих програма на које је утицала ова промена. Што је још горе при ажурирању се често праве грешке.

Редуданса података: Како се у приказаном систему процеси одвијају независно једни од других, понављање података није изузетак већ је правило. Због непотребних дупликата потребан је већи простор за њихово чување као и више труда и рада при њиховом ажурирању. Непланирана редуданса података може да доведе до губитка. На пример, исти подаци се могу водити под различитм именима атрибута у различитим документима, или обрнуто, исто име се може користити за различите врсте података.

Ограниченост дељења података: Коришћењем класичног система сваки процес има своје датотеке и корисници немају шансу да међусобно деле податке са корисницима из других процеса.

Дуго време за развој: Коришћењем класичног система заснованог на датотекама постоји мала шанса за коришћење претходних развојних достигнућа. Свака нова апликација захтева од пројектанта да крене од нуле. Сваки пут је неопходно дефинисати нове формате и описе података и писати код за приступ подацима за сваки програм. Овако велико време за развој није у складу са данашњим пословним променама, где је сваки минут битан да би се постигао успех.

Тешко одржавање прогрма: Скуп свих претходно наведених недостатака доводи до претеране потребе за одржавањем програма. Чак 80% буџета предвиђеног за развој система заснованог на датотекама одлази на његово одржавање. Због тога наравно, остаје јако мало простора за развој нових апликација.[3]

2.2 Приступ заснован на базама података

Приступ заснован на базама података(сервисни приступ) потенцира интеграцију и дељење података између свих одељења једне организације. Овај приступ захтева потпуну промену у начину размишљања, почевши од највишег нивоа управљања. Таква промена начина размишљања је за већину организација веома тешка.
Ако посматрамо претходно разматрани застарели информациони систем који се класично заснивао на датотекама. Одмах се може уочити да подаци који су претходно чувани у више различитих датотека, сада су интегрисани у јединствену базу података. Метаподаци, који описују ове податке, налазе се заједно са њима у бази података. Систем за управљање базама података пружа могућност стварања раличитих погледа на исту (или више) база података за различите кориснике у организацији. СУБП дозвољава корисницима да деле, претржују, приступају и ажурирају интегрисане податке. [3]

2.3 Предност приступа заснованог на базама података

Овај приступ има доста потенцијалних предности у односу на традиционални приступ. Те предности су следеће:

- **Независност између програма и података**
  Одвајање метаподатака од апликација која користе податке назива се независност података. Ова осбина код база података дозвољава промену и пренос података организације на друге рачунарске системе без потребе за променом програма који обрађује ове податке.

- **Минимална редуданса података**
  Циљ овог приступа је да се подаци који су се у претходном приступу чували одвојено (и због тога се више пута понављали) сада интегришу у јединствену логичку структуру. Сваки податак се налази само на једном месту у бази података. Приступ заснован на базама података не уклања редундансу у потпуности, али омогућава пројектанту базе података да пажљиво испланира врсту и количину редундансе.У неким случајевима је пожељно направити ограничену редундансу како би се перформансе базе података побољшале (нпр. убрзала претрага).

- **Побољшана конзистентност података**
  Елиминисањем (или контролисањем) редудансе података, у великој мери се смањују шансе за неконзистентност података. На пример, уколико је адреса купца записана на само једном месту не може да постоји неподударање у подацима у бази података. Такође, ажурирање података је у великој мери упрошћено када је свака вредност записана на само једном месту. На крају, уклањањем редудансе података долази до уштеде меморије.

- **Побољшана размена података**
  База података је дизајнирана као ресурс организације који користе сви њени запослени (којима је она неопходна у опису посла). Одређеним интернм и екстерним корисницима је дозвољено коришћење базе података, и сваки од њих (био у питању један корисник или група) има један или више погледа који му олакшавају коришћење базе подаака. Кориснички поглед је логички опис једног дела базе података који је неопходан кориснику да обави неки задатак.

- **Већа безбедност података**
  Када велики број корисника има приступ истим подацима постоји висок ниво ризика, да се подаци могу злоупотребити или да се чак могу неовлашћено мењати. Компаније улажу значајна средства, време и сопствене ресурсе у циљу обезбеђења својих података. СУБП је окружење преко кога се кроз функције администрације боље управља безбедносно осетљивим подацима и привилегијама крајњих корисника.

- **Повећана продуктивност у развоју апликација**
  Велика предност овог приступа је та што се у знатној мери смањују трошкови и време потребно за развој нових пословних апликација. Постоје два важна разлога зашто се апликације база података развијају знатно брже него код класичних система са датотекама.
    - Претпостављајући да су база података и све пропратне апликације већ направљене и имплементиране, програмер се може концентрисати на одређену функцију која је неопходна за нову апликацију, а не мора да размишља о дефинисању података или о детаљима везаним за имплементацију.
    - СУБП пружа велики број алата за извештавање. Као што су генератори форми и извештаја, и језика уз помоћ којих се аутоматизују неке од активности као што су дизајн и имплементација база података.

- **Смањена потреба за одржавањем програма**
  Сачувани подаци се морају често мењати из великог броја разлога: нове врсте података се додају, формати података се мењају, и тако даље. У систему обраде датотека, метаподаци и логика приступању подацима се налазе у индивидуалним апликационим програмима (ово је зависност између програма и података који су раније споменути). Као резултат овога, промена формата података и метода приступања моментално доводи до потребе мењања апликативних програма. Код база података, подаци знатно више независни од апликативних програма који их користе. У оквиру одређених граница, можемо да променимо једну од ставки, формат података или апликативни програм, а да не морамо да променимо другу ставку. Као резултат овога, јавља се смањење потреба за одржавањем програма. [3]

2.4 Трошкови и ризици приступа заснованог на базама података

У претодном делу текста наведено је неколико главних потенцијалних предности приступа заснованог на базама података. Међутим, код великог броја организација било је различитих проблема код остварења и искоришћења тих предности. На пример, постизање независности података (и стога, смањење потребе за одржавањем програма) се показало као тешко оствариво због ограничења старијих модела база података и софтвера за управљање базама података. На срећу, релациони модели (као и новији објектно-оријентисани модели) немају ових проблема. Други разлог за неуспех да се искористе ове предности је лоше планирање и имплементација база података – чак ни најбољи софтвери за управљање базом података не може да превазиђе овакве мањкавости. Приступ заснован на базама података садржи неке додатне трошкове и ризике који се морају решавати када се систем почне примењивати.

- **Ново,обучено особље**
  Често се дешава да предузеће, које се одлучи за приступ заснован на базама података, мора да ангажује или обучи људе за пројектовање, имплементирање и одржавање база података, као и да те људе укључи у постојећу радну организацију. Даље, због честих измена и развоја технологије, знање овог особља захтева сталну надоградњу и унапређивање. Једино оспособљено особље може да извуче максимум корисности из ових технологија.

- **Трошкови и сложеност инсталирања, управљања и рада система са базама података**
  Вишекориснички систем за управљање базама података је велики и сложен софтвер који у старту много кошта, захтева обучено особље за инсталирање и рад и има значајне годишње трошкове за одржавање и техничку подршку. Инсталирање оваквог система може захтевати надоградњу хардвера. Сталне обуке се подразумевају, да би се могле пратити нове верзије и ндоградње софтвера. Такође се може појавити потреба за додатним и скупљим софтвером за базе података ради веће сигурности података.

- **Трошкови прилагођавања(конвертовања) података**
  Термин наслеђени системи се углавном користи када се говори о старијим апликацијама у предузећу које су базиране на датотекама или старијим базама података. Трошкови прилагођавања оваквих старијих система за рад са модерним базама података (мерени у новцу, времену и обиму посла) често делује као велика препрека за предузеће.

- **Потреба за израдом сигурносних копија и опоравком података (*backup и restore*)**
  База података предузећа увек мора бити тачна и доступна. То захтева развијање и коришћење јасних процедура израде сигурносних копија као и опоравак базе података када неко оштећење настане. У данашњем окружењу, где постоје разноврсни безбедносни ризици, решавање овог проблема је од изузетне важности. Модеран систем за управљање базама података обично сам обавља израду сигурносних копија и опоравак података у случају хаварија. [3]

3. Безбедност информационих система

Да би заштитиле своје информатичке ресурсе, организације спроводе контролу безбедности или механизме одбране. Контрола безбедности треба да заштити све компоненте информационог система: податке, софтвер, хардвер и мрежу. Контрола треба да обезбеди спречавање случајних и намерних поступака који могу да оштете било који сегмент информационог система, да открије проблеме у раној фази и да их коригује, односно да унапреди опоравак од штете. Најефикаснија мера за постизање што боље контроле је едукација и обука корисника, чиме се постиже подизање свести запослених у организацији, о суштинској важности безбедности информационог система.

Различите апликације не решавају исте логичке проблеме на исти начин, иако се кориснику чини да се углавном понашају исто.
Због тог различитог приступа, злонамеран софтвер је писан с циљем напада на апликацију, тако да неку другу, са истом наменом неће оштетити. [2] [8]

3.1 ЗАШТИТА РАЧУНАРА

Када говоримо о заштити рачунара постоји више начина на који се ова област може посматрати. Пре свега, потребно је посветити пажњу физичком обезбеђењу рачунара где се мисли на обезбеђење од неовлашћеног приступа том рачунару тј. приступу одређеној просторији или згради у којој се налази рачунар. Под тим подразумевамо и заштиту од крађе, као и вандализма или саботаже.
Рачунар треба обезбедити од разних физичких оштећења услед пожара, поплаве, прегревања или електричних удара. Такође, веома је важно обезбедити исправан и несметан рад у екстремним условима као што су, на пример, високе температуре. Осим тога, на важним местима, нпр. у болницама, неопходни су и агрегати за независно и несметано напајање. И на крају, врло је важно имати и стручно особље које управља целим процесом и ради на рачунарима. [8]

3.2 ЗАШТИТА ПОДАТАКА

Заштита података је тема која данас постаје све значајнија услед све бржег технолошког развоја и све већег броја фактора који доприносе већој потреби за заштитом података. Некада је у фирмама било много мање рачунара те је и њихова заштита била много једноставнија, док данас у већим фирмама постоји и више стотина рачунара са различитим наменама и софтверима те је њихова заштита у многоме сложенија и тежа. Сам интернет представља опасност како за податке на Интернету, тако и за упаде путем Интернета.
Прављење резервних копија је најраспрострањенији начин заштите података. У основи, то је само копија најважнијих или свих података на рачунару. Подаци се чувају на хард дисковима, CD, DVD и осталим медијима.

Резервни подаци се требају чувати на више места ради боље безбедности. Такође, могуће је чувати податке на неком интернет серверу. Осим претње за безбедност рачунара, резервни подаци су такође битни ако се деси нека природна несрећа као што је земљотрес, пожар и тако даље. Резервни подаци су неопходани у случају напада вируса, када инфицирана датотека не може да се користи (мора да се обрише). [8]

3.3 ГУБИТАК ПОДАТАКА

Најчешћи узроци губитка података су највећим делом грешке корисника, односно запослених. Људски фактор је готово пресудан по овом питању, јер су рачунари ипак много савршенији и ретко сами направе грешку за разлику од човека. Затим, ту су и вируси, неовлашћен приступ, нестанак напајања, лоши резервни подаци и слично. На грешке које учини корисник може се утицати, тј. број грешака и губитак података се може смањити. То можемо урадити пре свега повећањем концентрације корисника тако што ћемо правити паузе сваких 15-так минута, затим бољим радним навикама као што су чешће прављење резервних *backup-a*, односно резервних копија, и наравно, бољом обученошћу корисника може се драстично смањити број грешака. Посебно треба водити рачуна о грешкама приликом снимања података, као и приликом брисања и копирања, јер су некада нетачни подаци исто што и изгубљени. Међутим, и овај проблем се лако превазилази прављењем копија, односно *backup* -овањем фајлова и докумената.

Најбољи начин заштите од губитка података је редовно прављење резервних копија. *Backup* података представља прављење копије фајлова, садржаја хард диска или дела садржаја хард диска, на неким медијима са којих се подаци могу поново рестаурирати односно повратити. Пре самог *backup*-а потребно је извршити одређене радње као припрему за *backup*. Треба прво извршити избор медија на коме ће се вршити *backup* (нпр. DVD, хард диск и сл). Треба изабрати софтвер из којег ће се вршити *backup* и одредити динамику прављења *backup*-а. Такође, пре прављења самог *backup*-а треба податке проверити односно скенирати на вирусе. Након извршеног *backup*-а пожељно је проверити исправност *backup*-ованих података.

Оно што је још важно је проверити начин именовања фајлова јер у неким варијантама *backup* софтвер неће допустити пролаз имена која нису *Windows* компатибилна. Треба поштовати конвенцију о именовању фајлова, и треба избегавати коришћење специјалних знакова као и ћириличних и латиничних слова у именима фајлова. Треба и правилно организовати податке на хард диску тј. не дозволити да се више садржаја повезаних међусобно налази разбацано у више различитих фолдера на диску. [3] [1]

4. Резервне копије (*backup*)

Као што је речено у претходном поглављу, мора постојати *backup* стратегија како би се минимизовао губитак података и повратили изгубљени подаци. Подаци се могу изгубити услед квара хардвера или софтвера или: [1]

- Случајном или злонамерном употребом DELETE упита
- Случајном или злонамерном употребом UPDATE упита - на пример, не коришћење WHERE клаузуле са UPDATE упитом (ажурирају се сви редови осим тачно одређених)
- Деструктивних вируса
- Природне катастрофе, као што су пожар, поплава, земљотреси
- Крађе

Ако постоји прикладна *backup* стратегија, могуће је повратити податке са минималним губитком времена и минималном шансом за трајни губитак података. *Backup* стратегија треба да врати систем у стање у ком се налазио пре појаве проблема.
Цена *backup* стратегије представља количину времена потрошену на дизајнирање, имплементацију, аутоматизацију и тестирање *backup* процедура.
Иако се губитак података не може спречити у потпуности, треба дизајнирати *backup* стратегију тако да се минимизира обим штете. При планирању *backup* стратегије, треба узети у обзир која је прихватљива количина времена у ком систем неће радити, као и прихватљива количина изгубљених података у случају системског квара.
Колико често је потребно правити резервне копије зависи од количине података које је прихватљиво изгубити и обима активности базе. [1]
При прављењу резервне копије базе података, треба размотрити следеће:

- Могуће је често прављење резервних копија базе ако се систем налази у *online transaction processing* (OLTP) окружењу
- Није потребно често прављење резервних копија базе ако је слаба активност система или се користи као подршка при одлучивању
- Треба заказати прављење резервних копија када SQL Server није окупиран ажурирањима
- Након одређивања *backup* стратегије, могуће је аутоматизовати процес коришћењем *Database Maintenance Plan Wizard*

4.1 Модели опоравка базе

Модел опоравка базе се може поставити или променити у било ком тренутку, али треба испланирати модел опоравка приликом прављења базе података.
SQL Server 2000 има три модела опоравка базе података. Сваки овај модел чува податке у случају отказивања сервера, али постоје битне разлике у томе како *SQL Server* враћа податке, како их складишти и које су потребне перформансе у случају квара. [1]

- **Потпуни модел опоравка** (*Full Recovery Model*) – овај модел се користити када потпун опоравак са оштећеног медија има приоритет. Овај модел користи копије базе података и све *log* информације за враћање базе података. *SQL Server* уноси све промене у базу података, укључујући масивне операције и прављење индекса. Узевши у обзир да *log* дневници нису оштећени, *SQL Server* може да поврати све податке осим трансакција које су се десиле у самом тренутку квара.

Пошто су све трансакције унете, опоравак се може покренути у било ком тренутку. *SQL Server 2000* подржава унос именованих ознака у *log* дневик како би омогућио опоравак тој одређеној ознаци.
Пошто ознаке *log* дневника заузимају простор у *log*-у, треба их користити само за трансакције које играју важну улогу у стратегији опоравка базе података.
Главно ограничење овог модела је велика величина *log* фајлова, резултујуће складиште и трошкови перформанси.

- **Масовно-логовани модел опоравка** (*Bulk_Logged Recovery Model*) – овај модел је сличан потпуном моделу опоравка, користи и базу података и *log* дневнике да би оправио базу података. Међутим, *Bulk_Logged Recovery model* користи *log* простор за следеће операције: CREATE INDEX, масивне операције учитавања, SELECT INTO, WRITETEXT и UPDATETEXT.
  Резултат чувања искључиво крајњих резултата вишеструких операција, log је мањи и масивне операције се могу брже покренути.
  Коришћење овог модела може да поврати све податке, али његова мана је немогућност повратка дела резервне копије, као што је враћање посебних ознака.

- **Једноставан модел опоравка** (*Simple Recovery model*) – обично се користи за мале базе података или базе података где се подаци ретко мењају. Овај модел користи потпуне или диференцијалне копије базе података и опоравак је ограничен враћањем базе података у стање у ком је била када је направљена последња резервна копија. Све промене настале након резервне копије су изгубљене и морају се поново унети. Основна предност овог модела је у томе што му је потребно мање складишног простора за *log* дневнике и што је најједноставинији за имплементацију.

SQL Server 2000 Standard Edition и *SQL Server 2000 Enterprise Edition*, подразумевано, користе потпун модел опоравка. Могуће је изменити модел опоравка у било ком треутку, али је потребно направити додатну резервну копију у тренутку промене. За откривање модела опоравка базе података, користи се DATABASEPROPERTYEX функција. [1]
Синтакса:

```
ALTER DATABASE database_name
SET RECOVERY {FULL | SIMPLE | BULK_LOGGED}
```

Следећи пример подешава да модел опоравка *Northwind* базе података буде Bulk_Logged модел опоравка.

```
ALTER DATABASE Northwind
SET RECOVERY BULK_LOGGED
```

4.2 Прављење резервних копија SQL Server-а

Током операција прављења резервних копија, *SQL Server*:

- Дозвољава прављење резервних копија базе док корисници настављају рад са базом података.
- Прави резервне копије оригиналних фајлова базе података и чува њихову локацију. Резервне копије садрже:
    - Схему и структуру фајлова
    - Податке
    - Делове фајлова *log* дневника. Делови *log* дневника који има резервну копију чувају активности базе података од почетка процеса прављења резервних копија.

SQL Server користи ове резервне копије за поновно прављење фајлова на њиховим оригиналнм локацијама, са свим објектима и подацима, приликом враћања базе.

- Чува активности базе које су се догодиле у процесу прављења резервних копија. [1]

4.3 Чување резервних копија

За прављење резервне копије базе података у *SQL Server-y*, мора се размотрити коме је дозвољено да прави резервне копије и где га складишти. Резервне копије базе података се могу направити извршавањем *Transact-SQL* упита или коришћењем *SQL Server Enterprise Manager*.

Чланови који имају следеће улоге могу правити резервне копије базе:

- Sysadmin има улогу у поправљању сервера
- db_owner има улогу у поправљању базе података
- db_backupoperator има улогу у поправљању базе података

Могу се направити и додатне улоге са дозволом за прављење копија базе података.
SQL Server може правити резервне копије на хард диску, траци или FIFO цеви. [1]

- Фајлови диска (локални или мрежни) су најчешће коришћен медиј за чување резервних копија
- Када се резервне копије чувају на траци, драјвер траке мора бити локално повезан са *SQL Server-ом.*

4.4 Када правити резервне копије

Када и колико често треба правити резервне копије базе података зависи од пословног окружења у ком се база података налази. Међутим, постоје тренуци када би било пожељно допунити *backup* стратегију. На пример, можда постоји потреба за повременим прављењем резервних копија системских база података или одређене корисничке базе. Иако *SQL Server* динамички прави резервне копије, неке активности се не могу појавити у бази података током операција прављења резервних копија. [1]

4.4.1 Системске базе података

Системске базе података складиште важне податке везане за *SQL Server* и све корисничке базе података. Самим тим, треба редовно правити резервне копије системских база података, поготво након измена.
Мастер база података садржи информације о свим базама података у *SQL Server-y*. Треба правити резервну копију мастер базе података кад год се креира нека корисничка база података. Ово омогућава лакши опоравак и враћање корисничких база података уколико дође до оштећења базе података. Након што је мастер база података поново направшена и њени подаци враћени, могуће је вратити резервне копије других системских база података као и референце на постојеће корисничке базе података. [1]
Када се изврши одређен упит или системска процедура, *SQL Server* аутоматски мења мастер базу података. Самим тим, треба направити резервну копију мастер базе података након извршавања:

- CREATE DATABASE, ALTER DATABASE или DROP DATABASE упита којим се креира, мења или уклања база података
- sp_logdevice системска процедура, којом се мења *log* дневник
- sp_addserver, sp_dropserver или sp_addlinkedserver системске процедуре, која додаје или уклања сервер
- sp_addmessage системска процедура

Треба правити резервне копије **msdb базе података** након њене измене, зато што msdb садржи информације око послова, упозорења и оператора које користи *SQL Server Agent*. Ако не постоји тренутна резервна копија msdb базе података, морају се поново правити све системске базе података у случају квара система и затим поново креирати сваки посао, упозорење и оператора.

Треба правити резервне копије **модел базе података** уколико дође до њене измене да би се уључила подразумевана конфигурација за све нове корисничке базе података. Зато што су корисничке базе података поново направљене при поновном прављењу мастер и msdb база података, промене модел базе података су такође изгубљене. Могуће је повратити модел базу података из резервних копија у случају квара система. [1]

4.4.2 Корисничке базе података

Треба редовно правити резервне копије корисничких база података. Поготово након прављења базе података или индекса или када је извршена одређена операција везана за лог дневник. [1]

- После креирања базе

Треба направити резервну копију базе података након њеног прављења или након убацивања података. Без потпуног прављења резервне копије базе података, не могу се повратити резервне копије *log* дневника зато што морају постојати смернице да би се знало на који се *log* дневник односи.

- После креирања индекса

Треба направити резервну копију базе података кад год се направи неки индекс. Иако није неопходно, у случају губитка базе података, сачуваћеш време током процеса опоравка. Прављење копија након направљеног индекса пружа гаранцију да ће резервна копија базе података садржати

структуру података и индекса. Ако се прави само резервна копија *log* дневника након креираа индекса, и у неком тренутку покушаш да повратиш тај *log* дневник, *SQL Server* ће морати да поново направи те индексе. Време потребно за поновно прављење може бити дуже него време потребно за опоравак потпуне резервне копије базе података.

- Након чишћења лог дневника

Треба направити резервну копију базе података након чишћења лог дневника упитом BACKUP LOG WITH TRUNCATE ONLY или BACKUP LOG WITH NO LOG. Када се изврше ови упити, лог дневник више неће садржати сачуване активности базе података и неће се моћи користити за враћање промена базе података.

- Након извршавања нелогованих операција

Операције које нису забележене у логу се називају нелоговане операције. У случају неких модела опоравка, није могуће повратити промене настале неким од следећих нелогованих операција

  - BACKUP LOG WITH TRUNCATE ONLY или BACKUP LOG WITH NO LOG упит. *SQL Server* уклања неактивне делове лог дневника без прављења резервних копија. Додатно, скраћивање лог дневника није снимљено у лог дневник.
  - WRITETEXT или UPDATETEXT упит. *SQL Server* мења податке у текст колонама и подразумевано, не записује ову активност у лог дневник. Како год, ове активности се могу записати у лог коришћењем опције WITH LOG
  - SELECT…INTO упит при креирању сталне табеле или масивног програма за прављење копија
  - Треба правити резервне копије базе података након извођења нелогованих операција, зато што ако систем застрани, лог дневник неће садржати потребне информације за враћање базе података у конзистентно стање.

4.5 Операције ограничене током прављења резервних копија

Могуће је направити резервну копију базе података док је база података онлине и активна. Међутим, неке операције нису препоручљиве током процеса прављења резервних копија. [1]
Избегавати следеће активности:

- Креирање или измена базе са CREATE DATABASE или ALTER DATABASE упитом.
- Извођење *autogrow* операција.
- Креирање индекса.
- Извођење било које нелогованих операција, укључујући масивно учитавање података и SELECT…INTO, WRITETEXT и UPDATETEXT упите.
- Проширивање базе података.

4.6 Прављење резервних копија

Када се праве резервне копије, прво се морају креирати *backup* фајлови (стални или привремени) У којим ће бити сачувана резервна копија. *SQL Server* обезбеђује опције за сваку од различитих метода прављења резервних копија. Иако *SQL Server* такође дозвољава да се изабере број дестинација на којим ће се физички чувати резервне копије, најчешће се резервне копије чувају на диску или траци. [1]

4.6.1 Креирање *backup* уређаја

За поновно користишћење *backup* фајлова који су направљени или за аутоматизацију прављења резервних копија базе података, потребно је креирати сталне *backup* уређаје. *Backup* уређаји се праве са *SQL Server Enterprise Manager* или извршавањем sp_addumpdevice системске процедуре. [1]

- Коришћење sp_addumpdevice процедуре за креирање *backup* уређаја на диску, траци или директно подотатке на FIFO цев. Кад се направе *backup* уређаји:
  - *SQL Server* креира логичка и физичка имена у sysdevices системску табелу мастер базе
  - Морају се спецификовати логичка и физичка имена *backup* фајла.
  - Може се направити до 64 *backup* фајла за базу података.

Када се креира нови *backup* уређај са *SQL Server Enterprise Manager-ом*, *SQL Server* извршава sp_addumpdevice системску процедуру.

Синтакса: sp_addumpdevice [@devtype =] 'device_type',
[@logicalname=]'logical_name',
[@physicalname=]'physical_name'
[,{ [@cntrltype =] controller_type | [@devstatus =] 'device_status'}]

Где је device_type: {DISK | TAPE | PIPE}

Пример1: У овом примеру је направљен стални *backup* фајл на хард диску.

```
USE master
  EXEC sp_addumpdevice 'disk', 'mybackupfile',
  'C:\ Backup\MyBackupFile.bak'
```

Пример2: У овом примеру је направљен *backup* уређај на траци са логичким именом Mytape1 и физичким именом \\.\tape0

```
USE master
EXEC sp_addumpdevice 'tape', 'mytape1', '\\.\tape0'
```

4.6.2 Прављење Backup фајлова без сталних уређаја

Док је прављење сталних *backup* уређаја препоручљивије, могу се правити и привремени *backup* фајлови BACKUP DATABASE упитом без потребе за спецификацијом *backup* уређаја. Ако се не планира поновна употреба *backup* фајлова, боље је креирати *backup* фајл без сталног уређаја. На пример, ако се прави резервна копија базе података искључиво за једно коришћење или се тестира одређена операција прављења резервне копије коју је потребно аутоматизовати, боље је направити привремен *backup* фајл. [1]

- Коришћење BACKUP DATABASE упита
  BACKUP DATABASE упитом се праве привремени *backup* фајлови. Пре него што *SQL Server* направи резервну копију, он прави *backup* фајл за чување резултата *backup* операције. Привремен *backup* фајл не сме постојати пре прављења резервне копије. Ако се прави привремен *backup* фајл, треба:
  - Одредити тип медија (диск, трака или цев).
  - Дефинисати комплетну путању до фајла

Синтакса:

BACKUP DATABASE {database_name | @database_name_var}
TO <backup_device> [, ...n]

Где је <backup_device> :

{{backup_device_name | @backup_device_name_var} | {DISK | TAPE | PIPE} = {'temp_backup_device' | @temp_backup_device_var}

Пример: У овом примеру се креира привремен *backup* фајл на диску и прави се резервна копија мастер базе у привремени *backup* фајл.

```
USE master
BACKUP DATABASE Northwind
  TO DISK = 'C:\Temp\MyCustomers.bak'
```

4.6.3 Коришћење вишеструких фајлова за чување копија

SQL Server може попуњавати више *backup* фајлова истовремено (паралелно). Када постоји неколико *backup* фајлова, подаци су издељени у све фајлове који су се користили за прављење резервних копија. Ови фајлови представљају *backup* сет.

Прављење резервних копија на вишеструким тракама или контролорима диска смањује време које је потребно за прављење резервних копија базе података. На пример, ако је за *backup* операцију која користи једну траку потребно 4 сата, може јој се додати друга трака и тиме ће се потребно време редуковати на само 2 сата. [1]

Синтакса

BACKUP DATABASE {database_name | @database_name_var}
TO <backup_device> [, ...n]
[WITH
[MEDIANAME = {media_name | @medianame_var}]]

Када се користе вишеструки фајлови за чување резервних копија, имати у виду следеће:

- Сви уређаји који се користе у појединачној *backup* операцији морају бити истог типа медија (диск или трака). Не могу се мешати диск и трака типови уређаја за појединачни медија сет. Медија сет је колекција фајлова која се користи за чување једног или више *backup* сетова.
- Може се користити комбинација сталних и привремених фајлова при прављењу *backup* сета.
- Ако се фајл дефинише као члан *backup* сета, фајлови се увек морају заједно користити. Не може се користити само један члан *backup* сета sa *backup* операцију осим ако се промени формат фајлова. Ако се промени формат једног члана *backup* сета, подаци који се налазе у осталим члановима *backup* сета постају неисправни и не могу се користити.

На пример, ако је *backup* сет направљен од два фајла, свака следећа *backup* операција која укључује исти *backup* сет мора користити иста та два фајла. Могу се додати додатне резервне копије тим фајловима. Међутим, ако је потребно користити један фајл од ових фајлова за опоравак базе података или за коришћење као дела неког другог сета, мора се променити формат фајла.

MEDIANAME опција дефинише име целог медија сета. Када се користе вишеструки фајлови за прављење резервних копија базе података, потребно је користити MEDIANAME опцију. MEDIANAME опција повезује вишеструке фајлове један са другим као члановима медија сета. Након што је медија сет направљен и именован, може се поново користити медија сет за будуће *backup* операције. Имена могу имати до 128 карактера.

4.6.4 Коришћење BACKUP упита

Backup операција се може извршити са *SQL Server Enterprise Manager*, *Backup Wizard* или *Transact-SQL*. [1]

Синтакса:

```
BACKUP DATABASE {database_name | @database_name_var}
TO <backup_device> [, ...n]
[WITH
    [FORMAT]
    [[,] {INIT | NOINIT}]
]
```

При прављењу резервне копије базе, треба одлучити да ли писати преко или додати у *backup* фајл:

- *SQL Server* ,подразумевано, додаје копију у фајл. Користећи NOINIT опцију, *SQL Server* додаје резервне копије у постојећи *backup* фајл или *backup* сет.
- Користећи INIT опцију, *SQL Server* пише преко постојећих податаке који се налазе у медија сету али задржава *header* информације.
- *Backup* операције греше и подаци нису прегажени ако:
- EXPIREDATE опција *backup* уређају још није истекла
- Backup_set_name параметри у опцији NAME се не поклапају са backup_set_name у *backup* уређају
- Покушаваш да прегазиш једног члана од претходно именованог *backup* сета. *SQL Server* детектује да ли је фајл члан *backup* сета
- Опција FORMAT се користи да прегази садржај *backup* фајла и да подели *backup* сет:
- Нови *header* је написан свим фајловима који су коришћени за ову *backup* операцију.
- *SQL Server* гази оба постојећа медија и садржај *backup* фајла.
- Користити FORMAT опцију пажљиво. Форматирање само једног *backup* фајла медија сета може да направи цео *backup* сет бескорисним.
  На пример, ако је трака која садржи део постојећег *backup* сета поново форматирана, цео *backup* сет је бескорисан.

4.6.5 Прављење резервних копија на траци

Траке су добар медијум за прављење резервних копија зато што су јефтини, обезбеђују велику количину складишта и могу се користити *off-site* за безбедност и поузданост података. [1]

Када се прави резервна копија базе података на траци, трака мора бити локално повезана са *SQL Server-ом*. *SQL Server* чува основне податке о копији на етикети таке, као што су:

- Име базе
- Време
- Датум
- Тип резервне копије

SQL Server користи стандардни формат за запис резервних копија на траку који се зове *Microsoft Tape Format*. Као резултат, и *SQL Server* и *non-SQL Server* подаци могу бити сладиштени на исту траку. На пример,и *SQL Server backup* и Microsoft Windows® 2000 *backup* могу постојати на истој траци.

Када се чувају резервне копије на траци, могу се користити опције које се примењују само за овај медијум.

- UNLOAD
- *SQL Server* се аутоматски празни траку након што је *backup* завршен. UNLOAD је подразумевана опција *SQL Server-a*.
- NOUNLOAD
  Ова опција се користи како *SQL Server* не би аутоматски брисао све са траке након *backup-a*.
- BLOCKSIZE
  Када се резервне копије чувају на траци, *SQL Server* одређује одговарајућу блок величину. Ова опција се користи за измену физичке блок величине у бајтовима.
- FORMAT
  Користи се за измену *headera* свих фајлова коришћених за *backup*.
- SKIP
  Користи се за прескакање *headera*. *SQL Server* игнорише све постојеће етикете траке. Етикете траке омогућавају упозорење везано за датум истека траке,...
- NOSKIP
  Користи се читање етикета траке. *SQL Server* ће проверити датум истека и имена свих *backup* сетова на медију пре писања преко ње. Ово је подразумевана опција.
- RESTART
  Користи се за рестарт *backup* операције од тренутка када је прекинута у току *backup* процеса. Мора се ручно рестартовати *backup* процес извршавањем оригиналног BACKUP упита са RESTART опцијом.

4.7 Методе прављења резервних копија

4.7.1 Метода потпуног прављења резервних копија

Ако је база података примарно *read-only* база података, потпуни *backup* базе података може бити довољан да спречи губитак података. [1]
При извођењу потпуног *backup-a* базе података, *SQL Server*:

- Прави се копија сваке активности која се одиграла током прављења резервне копије.
- Прави се копија неизвршених трансакција које се налазе у лог дневнику. *SQL Server* користи делове лог дневника које су сачуване у *backup* фајлу за обезбеђивање исправности података када се опоравак базе података десио.

Пример1: У овом примеру се прави именовани *backup* уређај са логичким именом Nwindbac и изводи се потпуно прављење резервних копија.

```
USE master
EXEC sp_addumpdevice 'disk', 'NwindBac', 'D:\MyBackupDir\NwindBac.bak'
BACKUP DATABASE Northwind TO NwindBac
```

Пример2: У овом примеру се изводи потпуно прављење резервних копија у NwindBac file и пише се преко претходне верзије тог фајла.

```
BACKUP DATABASE Northwind TO NwindBac WITH INIT
```

Пример3: У овом примеру се изводи потпуно прављење резервних копија у NwindBac фајл. Сви

претходни фајлови су недирнути.

```
BACKUP DATABASE Northwind TO NwindBac WITH NOINIT
```

Пример4: У овом примеру се креирају *backup* фајлови на диску и изводи се потпуно прављење резервних копија на том фајлу.

```
BACKUP DATABASE Northwind TO
DISK = 'D:\Temp\MyTempBackup.bak'
```

4.7.2 Прављење дифернцијалних резервних копија

Извођење диференцијалног прављења резервних копија треба да минимизује време потребно за опоравак базе података која се често мења. Треба изводити диференцијални *backup* само ако је потпуни *backup* базе података већ извршен. [1]
Приликом дифернцијалног прављења резервних копија, *SQL Server*:

- Прави резервну копију делова базе података које су се промениле од последњег потпуног *backup-a* базе података.
- Прави резервну копију сваке активности која се десила током диференцијалног *backup-a*, као и сваке неизвршене трансакције у лог дневнику.
- Када се изводи диференцијални *backup*, узети у обзир:
- Ако је одређени ред у бази података измењен неколико пута након последњег потпуног *backup*-а базе података, диференцијални *backup* садржи само последњи сет вредноти тог реда. Ово је другачије од *backupa* лог дневника који садржи историју свих измена тог реда.
- Смањује време потребно за *backup* базе података зато што су *backup* сетови мањи него у потпуном *backupu*.
- Минимизује време потребно за враћање базе података зато што не мора да се пријављује серија измена лог дневника.
- Треба одредити конвецију за давање имена *backup* фајловима које садрже диференцијалне резервне копије за разликовање њих од фајлова који садрже потпуни *backup* базе података.

Синтакса:

```
BACKUP DATABASE {database_name | @database_name_var}
 TO <backup_device> [, ...n]
 [
WITH
   [DIFFERENTIAL]
]
```

Пример: У овом примеру се прави диференцијални *backup* у привременом *backup* фајлу.

```
BACKUP DATABASE Northwind TO
DISK = 'D:\MyData\MyDiffBackup.bak'
 WITH DIFFERENTIAL
```

4.7.3 Прављење резервних копија лог дневика

Праве се резервне копије лог дневника да би се чувала свака промена базе података. Обично се прави резервна копија лог дневника када се изводи потпуни *bakup* базе података: [1]
Не треба правити резервну копију лог дневника, ако не постоји барем једна потпуна резервна копија.

- Лог дневник не може бити враћен без одговарајуће резервне копије базе података.
- Лог дневник не може бити враћен ако се користи Једноставан Модел Опоравка.

Када се прави резервна копија лог дневника, *SQL Server*:

- Прави резервну копију лог дневника из последњег успешно извршеног BACKUP LOG упита.
- Лог дневник се скраћује све до почетка активног дела лог дневника и одбацују све информације неактивног дела.

Синтакса:

```
BACKUP LOG {database_name | @database_name_var}
TO <backup_device> [, …n]
[WITH
   [{INIT | NOINIT}]
]
```

Пример: У овом примеру се креира *backup* уређај и прави се резервна копија лог дневника Northwind базе података.

```
USE master
EXEC sp_addumpdevice 'disk', 'NwindBacLog',
    'D:\Backup\NwindBacLog.bak'
BACKUP LOG Northwind TO NwindBacLog
```

Ако су фајлови базе података оштећени или изгубљени, треба направити резервне копије лог дневника са NO_TRUNCATE опцијом. Коришћењем ове опције праве се резервне копије свих недавних активности базе података.
SQL Server:

- Чува цео лог дневник (све што се десило од поледњег BACKUP LOG упита) иако се бази података не може приступити.
- Не чисти лог дневник од извршених трансакција.
- Дозвољава опоравак података када је систем отказао.

Када се поврати база података, може се повратити *backup* базе података и применити резервна копија лог дневника који је направљен са NO_TRUNCATE опцијом за опоравак података.
Може се користити BACKUP LOG упит са TRUNCATE_ONLY или NO_LOG опцијом за чишћење лог дневника. Треба правити резервну копију лог дневника редовно да би се одржавала нормана величина лог дневника:

- Ако је лог дневник пун, корисници не могу ажурирати базу података и не може се потпуно опаравити база у случају отказа система. Мора се очистити лог дневник или извођењем потпуног *backupa* базе података и сачувати податке или скраћивањем лог дневника.
- Ако прављење резервне копије лог дневника не скрати величину лог дневника онда је могуће да постоји стара трансакција која је отворена у лог дневнику.

TRUNCATE_ONLY и NO_LOG опције изводе исту функцију. Ако треба очистити лог дневник и не треба чувати резервне копије података, треба користити ове опције. *SQL Server* уклања неактивне делове лог дневника без прављења њихових копија. Активни делови лог дневника који се састоје од неизвршених трансакција се никад не скраћују.

Синтакса:

```
BACKUP LOG {database_name | @database_name_var}
[WITH {TRUNCATE_ONLY | NO_LOG }]
```

Пример1: У овом примеру се користи BACKUP LOG упит за уклањање неактивних делова лог дневника без прављења резервних копија.

```
BACKUP LOG Northwind WITH TRUNCATE_ONLY
```

Пример2: У овом примеру се користи BACKUP LOG упит за уклањање неактивних делова целог лог дневника без прављења *backup* копија њега.

```
BACKUP LOG Northwind WITH NO_LOG
```

Може се подесити trunc. log on chkpt. опија на *true* за испис свих извршених трансакција базе података када се појави контролна тачка. Ова опција аутоматски скраћује лог дневник.

4.7.4 Прављење резервних копија фајла или групе фајлова базе података

Ако се изводи потпуни *backup* базе података на веома великој бази података (*VLDB - very large databases*) која није у деловима, треба извршити *backup* фајла или групе фајлова. Када *SQL Server* прави резервне копије фајла или групе фајлова, он: [1]

- Прави резервне копије само фајлова базе података који су дефинисани у FILE или FILEGROUP опцији.
- Дозвољава прављење резервних копија одређених фајлова базе података уместо целе базе.

Када се извршава *backup* фајла или групе фајлова, онда:

- Морају се дефинисати фајлови или групе фајлова.
- Мора се извршити *backup* лог дневника.
- Треба утврдити план за *backup* сваког фајла у кружном смеру.
- Може се дефинисати до 16 фајлова или група фајлова.

Синтакса:

```
BACKUP DATABASE {database_name | @database_name_var}
[ <file or file group> [, ...m]] TO <backup_device> [, …n]]
```

Где је <file or file group>:

```
{FILE = {logical_file_name | @logical_file_name_var}
| FILEGROUP = {logical_filegroup_name | @logical_filegroup_name_var} }
```

Пример: У овом примеру се прави резервна копија фајла Orders2 из групе фајлова. PhoneOrders база података састоји се из 3 фајла: Orders1, Orders2 и Orders3. Лог дневник се складишти у Orderlog фајл. Ови *backup* фајлови већ постоје: OrderBackup1, OrderBackup2, OrderBackup3 и OrderBackupLog.

```
BACKUP DATABASE PhoneOrders
FILE = Orders2 TO OrderBackup2
 BACKUP LOG PhoneOrders to OrderBackupLog
```

4.8 Стратегије планирања прављења резервних копија

Када се планира стратегија прављања резервних копија, пословно окружење у ком се налази база података одређује метод прављења резервних копија или комбинације метода које треба изабрати. [1]

4.8.1 Стратегија потпуног прављења резервних копија

Величина базе и колико често се мењају подаци у њој одређују време и ресурсе који су укључени у имплементацију стратегије потпуног прављења резервних копија.
Извршити потпуно прављења резервних копија базе података ако:

- База је мала. Време потребно за прављење резервних копија мале базе је прихватљив.
- База има само неколицину измена или је *read-only*. Извршавање потпуног прављења резервних копија базе података обухвата разуман, комплетан сет података. Уколико је прихватљив мањи губитак података уколико база откаже између прављења резервних копија.

Ако се имплементира само стратегија потпуног прављења резервних копија базе података, лог дневник ће се временом напунити. Када лог дневник постане пун, *SQL Server* може спречити даље активности базе док се не очисти лог дневник. Треба периодично чистити лог дневник.
Може се подесити trunc. log on chkpt. опција на *true* за минимизовање величине лог дневника.
При употреби ове опције, све извршене трансакције су написане у бази када се појави контролна тачка и лог дневник је аутматски скраћен.

Пример1 стартегије

Претпоставке су следеће:

- База има само 10 мегабајта података.
- За потпуно прављење копије базе података је потребно неколико минута.
- База се већински користи као помоћ при одлучивању и веома ретко се мења у току дана.
- Могућност губитка дневних промена базе је прихватљив. Ове промене се могу лако рекреирати.
- Систем администратор не жели да прати величину лог дневника нити да одржава лог дневник.
- Trunc. log on chkpt. опција базе података је постављена на *true* да би осигурала да је лог дневник повремено скраћен (очишћен). Лог дневник се не користи за чување промена базе и не може се користити за опоравак базе у случају квара система.
- Потпуно прављење резервних копија се врши свако вече у 18h.
- База се квари у 10h.

За опоравак базе, треба успоставити потпуно направљену резервну копију од последње вечери у 18h, писаће преко покварене верзије базе. Ограничење оваквог приступа је што ће све измене направљене од 18h бити изгубљене.

Пример2 стратегије

Претпоставити да је база слична описаној у Примеру1 са следећим изузецима:

- База се јако ретко мења у току дана (али мало чешће него база из Примера1).
- Систем администратор жели да се осигура да постоји адекватан простор у лог дневнику.
- Trunc. log on chkpt. опција је обрисана (подешена на *false*). Лог дневник чува промене настале од последњег потпуног прављења резервних копија и може се користити за обнову базе ако

систем откаже.
- Лог дневник се чува у одвојеном физичком уређају у односу на базу података.
- Потпуно прављење резервних копија се врши сваки дан у 18h. Прављење резњервне копије лог дневника се не изводи свакодневно, али се лог дневник чисти повремено.

За опоравак базе је потребно испратити следеће кораке:
1. Прављење резервне копије лог дневника без скраћивања података (NO_TRUNCATE опција).
2. Успостављање потпуне резервне копије од претходне ноћи 18h.
3. Обнавља резервну верзију лог дневника који је направљен у кораку 1, и опоравља базу податка.
Користећи овај приступ, постоји могућност повратка измена насталих након *backupa* од претходне ноћи ако лог дневник није оштећен. Међутим, ако су потенцијално изгубљени подаци превише велики, потребно је размотрити имплементирање стратегије опоравка која укључује периодично прављење резервних копија лог дневника.

4.8.2 Стратегија потпуног прављења резервних копија базе података и лог дневника

Као додатак потпуног прављења резервних копија базе података, такође је потребно направити и резервну копију лог дневника да би постојали снимци свих активности базе који су се појавили између потпуних *backup-ова* базе података. Ово је честа backup стратегија.
Када се имплементира стратегија потпуног прављења резервних копија базе података и лог дневника, може се обновити база података из најскорије потпуне резервне копије базе података и тада применити све трансакције из резервне копије лог дневника који су креирани од последњег потпуног *backup* базе података.

Изводити стратегију потпуног прављење резервних копија базе података и лог дневника за базе података које се често мењају. Такође узети у обзир да ли се прављење копија базе података и лог дневника може одвити у прихватљивом периоду времена.

Пример стратегије

Претпоставити следеће:
- База података и лог дневник се чувају у одвојеним фајловима у различитим физичким медијима.
- Потпуно прављење резервних копија базе података се врши свако вече у 18h.
- Прављење копија лог дневника се врши сваки дан у 9h, 00:00 i 3h
- Физички медиј који садржи квар базе десио се у 13h.

За опоравак базе података је потребно:
1. Направити резервну копију лог дневника, ако је могуће. Користити WITH NO_TRUNCATE опцију.
   2. Успоставити потпуну резервну копију базе података која је направљена претходно вече у 18h.
   3. Применити све резервне лог дневнике који су направљени тог дана (9 - 12h).
   4. Применити копију лог дневника који је направљен на почетку процеса опоравка базе подататака (уколико је направљен).

4.8.3 Стратегија диференцијалног прављења копија

При имплементацији стратегије диференцијалног прављења резервних копија, мора се укључити потпуни *backup* базе података, као и лог дневника. Диференцијални *backup* се састоји само од делова базе која се мењала од последњег потпуног прављења резервних копија базе података.
У стратегији диференцијалног прављења копија, *SQL Server*:

- Не обухвата промене у лог дневнику. Самим тим, треба повремено направити резервну копију лог дневника.
- Захтева да се обнови само последњи диференцијани *backup* за опоравак базе података.
- Последњи диференцијални *backup* садржи све промене које су направљене у бази од последњег потпуног *backup* базе података.

Ова стратегија се користи за смањење времена оправка ако је база оштећена.

Пример стратегије

Претпоставити следеће:

- Потпуно прављење резервне копије базе података се врши једном недељно. Последњи пут је направљен у недељу у 1h.
- Диференцијално прављење резервних копија се врши на крају сваког радног дана. Урађен је и у понедељак и у уторак у 18h.
- Прављење резервне копије лог дневника се врши сваког сата током радног дана (од 8 - 17 h). Последњи пут се извршио у среду у 8h и поново у 9 h.
- База се покварила у среду у 9:30.

За опоравак базе треба:
1. Направити резервну копију лог дневника, ако је могуће. Користећи WITH NO_TRUNCATE опцију.
2. Успоставити потпуну резервну копију базе података која је направљена у недељу у 1h
3. Успоставити диференијалну резервну копију која је направљена у уторак 18h. Пошто је то последња диференцијална копија и садржи све промене направљене у бази од последњег потпуно направљене копије базе података од недеље у 1h
4. Применити резервну копију лог дневник која је направљена у среду у 8h и 9h.
Применити резервну копију лог дневник која је направљена на почетку процеса опоравка (Корак 1) да би се обезбедила исправност података.

4.8.4 Стратегија прављења резервне копије фајла или групе фајлова

При имплементацији стратегије прављења резервне копије фајла или групе фајлова, обично се прави копија лог дневника као део стратегије.

Користи се ова стратегија за *VLDB* која је издељена у велики број фајлова. У комбинацији са регуларним прављењем резерних копија лог дневника, ова техника нуди временски осетљиву алтернативу за потпуно прављење резерввних копија базе података.
Ова стратегија је компликована и не успоставља референцијални интегритет.

Пример стратегије

Претпоставити следеће:

- Подаци у бази су подељени у: Филе1, Филе2 и Филе3.
- Потпуно прављење резервних копија базе се врши сваке недеље. Последњи пут се извршио у понедељак у 1h.
- Направљене су резервне копије фајлова у кружном смеру сваки дан у 1h
    - Филе1 је у уторак у 1h.
    - Филе2 у среду у 1h.
    - Филе3 у четвртак 1h.
- Прављење копије лог дневника се врши сваки дан у 00:00 и 18h.
- У четврток у 8h, физички медијум Филе2 се квари.

За опоравак базе података треба:
1. Направити резервну копију лог дневника, ако је могуће. Користећи WITH_ NO_TRUNCATE опцију.
2. Успоставити копију Филе2 која је направља у среду у 1:00 A.M.
3. Применити све резервне копије лог дневника који су направљени од среде у 1h.
4. Применити лог дневник направљен на почетку процеса опоавка података. Применом свих лог дневника праве се објекти Филе2 конзистентни са остатком базе.

Перформансе добијене коришћењем ове стратегије су резултат чињенице да су само примењени догађаји лог дневника који погађају податке који су сачувани у Филе2. Догађаји лог дневника пре 1h од среде се не користе. Само трансакције Филе2 после 1h среде се користе.

5. Процес опоравка базе података

Процес опоравка у *SQL Server-y* је унутрашњи механизам који обезбеђује да база буде конзистентна испитујући лог дневник и предузимајући одговарајуће акције: [1]

- *SQL Server* испитује лог дневник од последње контролне тачке до тренутка када је *SQL Server* отказао или се искључио. Контролна тачка је као обележивач, који означава место где су све промене података записане у базу.
- Ако лог дневник има извршене трансакције које још увек нису унете у базу, *SQL Server* преноси трансакције, пријављујући промене у базу података.
- Ако лог дневник садржи неку неизвршену трансакцију, *SQL Server* поништава ове трансакције. Неизвршене трансакције неће бити унете у базу.

Када је систем рестартован након отказа или гашења система, *SQL Server* започиње аутоматски процес опоравка да би обезбедио конзистентност података. Нема потребе за започињањем овог процеса ручно пошто ће се аутоматски започети.
Ручно се може иницирати процес опоравка када се изводе операције обнављања. Процес опоравка који је тако започет је сличан аутоматском процесу опоравка који се јавља када се *SQL Server* рестартује.
Када се обнавља база података, *SQL Server* аутоматски извршава одређене акције за осигурање да је база успостављена брзо и са минималним утицајем на продуктивност.

SQL Server извршава безбедносну проверу при извршавању RESTORE DATABASE упита. Овај интерни механизам спречава да се дође до случајног писања преко постојаће базе података са резервном копијом друге базе података или са непотпуним информацијама.

SQL Server не обнавља базу ако:

- База која је именована у RESTORE DATABASE упиту већ постоји на серверу, и име базе које је снимљено у резервне копије се разликује од базе која је именована у RESTORE DATABASE упиту.
- Сет фајлова базе података на серверу се разликује од сета фајлова базе података који се налази у *backup* сету.
- Сви фајлови или групе фајлова који су потребни за обнављање базе података нису обезбеђени.

На пример, при покушају да се успостави резервна копија Nortxwind базе података у базу која се зове Accounting, и Accounting, већ постоји на серверу, *SQL Server* ће спречити успостављање. За писање резервне копије Nortxwind преко података у Accounting, мора се прегазити безбедносна провера.

Када се обнавља база података од потпуне резервне копије, *SQL Server* поново ствара оригиналне фајлове базе података. И смешта их на локацију где су били снимљени када је направљена резервна копија. Сви објекти базе се аутоматски поново праве. Не мора се поново правити схема базе.

5.1 Припрема за обнављање базе података

Треба верификовати резервне копије ради потврде да се успостављају намеравани подаци и објекти и да копије садрже исправне информације. Пре него што се успоставе резервне копије, морају се извршити одређени задаци који омогућавају почетак процеса обнављања.

- Верификација резервних копија
  Пре него што се успостави *backup* фајл, мора се осигурати да је валидно и да фајл садржи очекиване копије. Може се користи *SQL Server Enterprise Manager* за преглед листе особина за сваки *backup* уређај. За детаљније информације везане за резервне копије, могу се извршити следећи упити.

  - RESTORE HEADERONLY упит
    Овај упит се користи за добијање *header* информација одређеног *backup* фајла или сета. Када се извршава RESTORE HEADERONLY упит, добијају се следеће информације:
    - Име и опис *backup* фајла или сета
    - Који је *backup* медиј коришћен, као што је трака или хард диск
    - *backup* метод,
    - Датум и време када се прављење копије извршило
    - Величина backup-а

  - RESTORE FILELISTONLY упит
    Овај упит се користи за добијање информација о оригиналној бази података или фајловима лог дневника који се налазе у *backup* фајлу. Извршавањем овог упита може се избећи обнављање погрешних *backup* фајлова.
    Када се извршава RESTORE FILELISTONLY упит, *SQL Server* враћа следеће информације:
    - Логичка имена базе података и фајлова лог дневника
    - Физичка имена базе података и фајлова лог дневника
    - Тип фајла, као што је база или лог дневник
    - Припадност групи фајлова
    - Величина *backup* сета, у мегабајтима (MB)
    - Максимална дозвољена величина фајла у MB

  - RESTORE LABELONLY упит
    Користи се за добијање информација о *backup* медију на ком се чува *backup* фајл.

  - RESTORE VERIFYONLY упит
    Користи се за верификацију појединачних фајлова *backup* сета, проверава да ли су сви читљиви и комплетни. *SQL Server* не верификује структуру података.

5.2 Обнављање базе података

Може се користити *SQL Server Enterprise Manager* или RESTORE упит и дефинисати опције које су специфичне за тип резервних копија које треба успоставити. [1]

- Коришћење RESTORE упита
  Овај упит се може користити за извршавање опрација обнављања. Овом опцијом је могуће дефинисати детаље о начину на који ће се резервне копије обновити.
  Синтакса:

```
RESTORE DATABASE {database_name | @database_name_var}
[FROM <backup_device> [, ... n]]
[WITH
   [FILE = file_number]
 [[,] MOVE 'logical_file_name' TO 'operating_system_file_name']
[[,] REPLACE]
 [[,] {NORECOVERY | RECOVERY | STANDBY = undo_file_name}]]
[[.] RESTART]
Где је backup_device> :
 {{backup_device_name | @backup_device_name_var} |
 {DISK | TAPE | PIPE} = {'temp_backup_device' | @temp_backup_device_var}
 }
```

Пример: У овом примеру треба обновити Northwind базу података из сталног *backup* фајла.

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac
```

Ако је база оштећена, коришћењем RESTORE упита се поново прави база података. Када се поново прави база података:

- Не треба уклањати оштећену базу података.
- *SQL Server* аутоматски рекреира фајлове и објекте базе података.

5.3 Иницирање процеса опоравка

Увек треба специфицирати RECOVERY или NORECOVERY опцију за спречавање административних грешки током процеса обнављања и за прављење упита једноставних за читање. RECOVERY је подразумевана опција *SQL Server-a*. Користи се са последњом копијом лог дневника за обнављање или са потпуном базом података за враћање у конзистентно стање: [1]

- *SQL Server* поништава све неизвршене трансакције који се налазе у лог дневнику, а процесира све извршене трансакције.
- База је доступна за употребу након што је процес опоравка завршен.

NORECOVERY опција се користи када постоје вишеструке резервне копије за обнављање.
Уколико се користи NORECOVERY опција:

- SQL Server нити поништава неизвршене трансакције у лог дневнику, нити процесира извршене трансакције.
- База података је недоступна док се процес опоравка не заврши.

5.4 Обнављање различитих типова резервних копија

За обнављање базе података, мора се знати тип методе прављења резервних копија и да ли резервна копија постоји. Треба утврдити да *backup* фајлови садрже копије које треба обновити. Уверити се да су резервне копије валидне и да су сви ти фајлови на локацији. [1]

5.4.1 Обнављање потпуне резервне копије базе података

Када се обнавља база података из потпуне резервне копије базе података, *SQL Server* рекреира базу података и све фајлове везане за њу и онда их смешта на оригиналну локацију. Сви објекти базе су аутоматски поново направљени.

Обнављање из потпуне резервне копије базе података се врши када:
- Физички диск базе података је оштећен.
- Цела ба за је оштећена, покварена или обрисана.
- Идентична копија базе података се обнавља на неки други *SQL Server*.

Пример: Овај пример предпоставља да постоји потпуна резервна копија у сталном NwindBac *backup* фајлу и да су 2 резервне копије додате том фајлу. Northwind база је потпуно измењена другом копијом на NwindBac *backup* уређај. Процес опоравка враћа базу у конзистентно стање (процесира извршене промене и поништава неизвршене активности)

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac WITH FILE = 2, RECOVERY
```

5.4.2 Обнављање диференцијалне резервне копије

Када се база података обнавља из диференцијалне резервне копије, *SQL Server*:
- Обнавља само делове базе података који су се изменили од последњег потпуног прављења резервне копије.
- Враћа базу података у стање у ком је била када се диференцијално чување резервних копија завршило.
- Често је потребно мање времена за обнављање диференцијалне резервне копије него за пријаву серије трансакција лог дневника који представљају исту активност базе података.

Пример: У следећем примеру је обављено диференцијално обнављање резервне копије без опоравка базе. *SQL Server* дозвољава да се успостави трансакција пре него што се Northwind врати у конзистентно стање дефинисајући NORECOVERY опцију.

```
USE master
 RESTORE DATABASE Northwind
 FROM NwindBacDiff
WITH NORECOVERY
```

5.4.3 Обнављање резервне копије лог дневника

Када се обнавља резервна копија лог дневника, *SQL Server* враћа промене базе података које су сачуване у лог дневник. Обично се обнавља лог дневник да би се примениле промене које су направљене у бази података од последњег потпуног или дифренцијалног прављења резервне копије. Могуће је обновити лот дневник да би се опоравила база података у одређеном временском тренутку.

Иако обнављање диференцијалних резервних копија убрзава процес обнављања, да би се обезбедила конзистентност података потребно је обновити, додатно, резервну копију лог дневника која је креирана после диференцијалног *backup*. Пре обнављања копије лог дневника, потребно је обновити потпуну резервну копију базе података. Када постоје вишеструки лог дневници за примену, дефинисање NORECOVERY опције за све лог дневнике осим последњег. *SQL Server* суспендује процес опоравка све док се последњи лог дневник не обнови.

Синтакса:

```
RESTORE LOG {database_name | @database_name_var}
[FROM <backup_device>[, …n]]
[WITH [{NORECOVERY | RECOVERY | STANDBY = undo_file_name}]
 [[,] STOPAT = {date_time | @date_time_var}]
 [[,] STOPBEFOREMARK = mark_name [AFTER date_time]
[[,] STOPATMARK = mark_name [AFTER date_time]
```

Пример: У овом примеру постоји потпуна резервна копија базе података у једном *backup* фајлу и 2 резервне копије лог дневника постоје у другом *backup* фајлу. Три одвојене операције обнављања су извршене да би се осигурала конзистентност базе података.

1. Први корак обнавља потпуну резервну копију базе података.

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac
WITH NORECOVERY
```

2. Други корак обнавља прву резервну копију лог дневника без опоравка базе.

```
USE master
RESTORE LOG Northwind
 FROM NwindBacLog
WITH FILE = 1,
STATS,
NORECOVERY
```

3. Трећи корак обнавља други лог дневник, процесира сваку извршену трансакију и одбацује сваку не изврешену трансакцију. RECOVERY опција враћа Northwind базу у конзистентно стање.

```
USE master
 RESTORE LOG Northwind
FROM NwindBacLog
WITH FILE = 2,
RECOVERY
```

5.4.4 Обнављање резервне копије фајла или групе фајлова

Ако постоји резервна копија фајла или групе фајлова, могуће их је обновити:
- Редукује време потребно за обнављање веома велике базе података (*VLDB*).
- Опоравља подаке када је одређени фајл случајно обрисан или оштећен.

Када се обнавља из фајла или групе фајлова, мора:
- Пријавити све лог дневнике који су креирани од када је креирана резервна копија фајла како би дошло до обнављања фајла или групе фајлова у стање које је конзистентно са остатком базе података. *SQL Server* пријављује само трансакције из лог дневника које утичу на дневнике.
- Обновити резервне копије фајлова као целине ако табела и индекси везани за њу постоје у две различите групе фајлава.

Синтакса:

```
RESTORE DATABASE {database_name | @database_name_var} [, ...m]
[FROM <file_or_filegroup> [, …n]]
Gde je < file_or_filegroup >
{FILE = logical_file_name |
FILEGROUP = logical_filegroup_name}
```

Пример: У овом примеру база података се сатоји од 3 фајла: Nwind1, Nwind2 и Nwind3. Nwind2 фајл садржи једну табелу и одговарајуће индексе. Nwind2 фајл је сачуван у Nwind2bac *backup* фајл. Једна резервна копија лог дневника је од Nwind2. Nwind2 мора бити обновљен зато што је физички медиј оштећен.

1. Први корак обнавља резервну копију Nwind2 базе података.

```
USE master
RESTORE DATABASE Northwind
     FILE = Nwind2
 FROM Nwind2Bac
WITH NORECOVERY
```

2. Други корак обнавља резервну копију лог дневника, процесирајући сваку извршену трансакцију, и одбацујући сваку неизвршену трансакцију.

```
 USE master
RESTORE LOG Northwind
FROM NwindBacLog
 WITH RECOVERY
```

5.4.5 Обнављање оштећене системске базе података

Ако је медиј који садржи системску базу података оштећен, биће потребно поново направити системске базе података.
Ако *SQL Server* сервис може да се покрене, може се користити RESTORE DATABASE упит или *SQL Server Enterprise Manager* за обнављање системских база података из резервних копија. [1]

Ако је мастер база података оштећена и ако се не може покренути *SQL Server*, треба извршити следеће кораке:
1. Поново направити системску базу података.
2. Рестартовати *SQL Server* сервис.
3. Успоставити резервне копије системских база података.

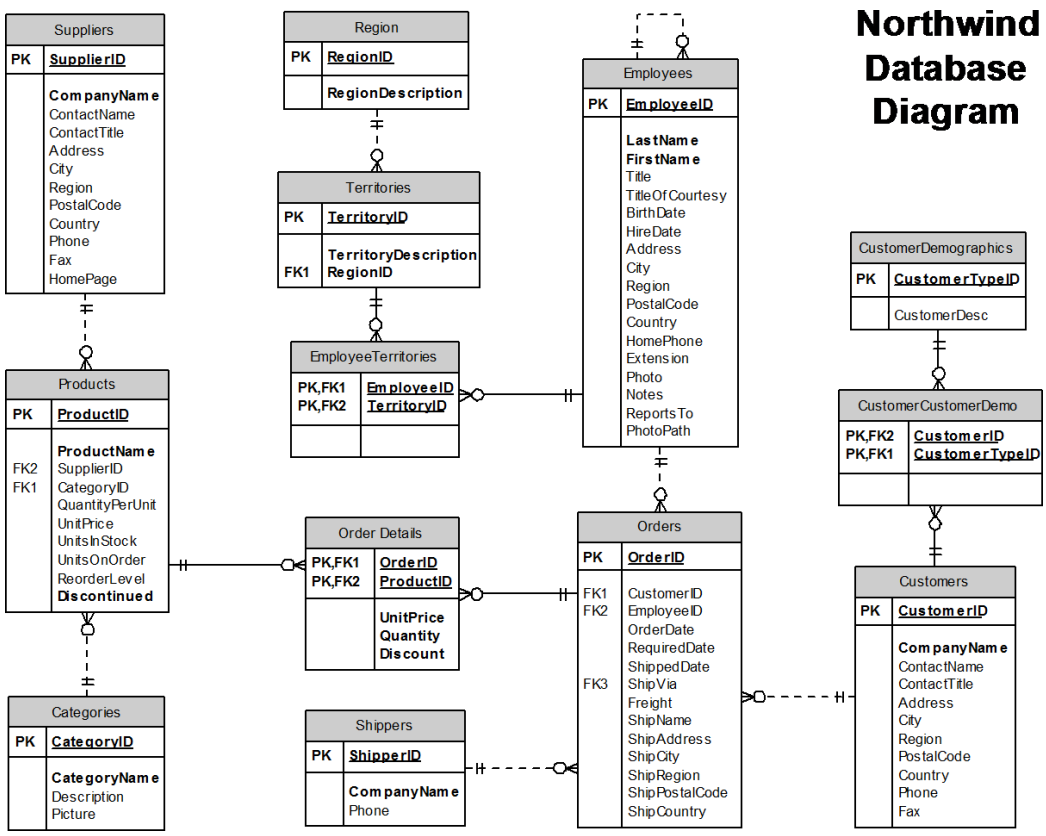
Након што су ситемске базе података поново направљене и SQL Server поново покренут, треба извршити следеће кораке:
1. Успоставити мастер базу података из резервне копије, ако постоји. Ако валидна резервна копија мастер базе не постоји, морају се рекреирати информације које су сачуване у мастер базу података ручно.
2. Успоставити msdb базу података из резервне копије ако постоји. Мора се успоставити msdb база података када се поново направи мастер база података.
3. Успоставити модел базу података, ако постоји.

Могуће је додати или обновити корисничке базе података у зависности од тога да ли је мастер база података поново направљена:

- Ако је мастер база података обновљена из валидне резервне копије, садржаће референце ка свакој корисничкој бази података. Даље акције неће бити потребне.
- Ако је мастер база података направљена и исправна резервна копија базе није примењена, мора се извести следеће:
    - Обновити корисничке базе података из резервне копије
    - Додати постојеће фајлове корисничких база података новој мастер бази. Ако фајлови нису оштећени додати их мастер бази коришћењем sp_attach_db или sp_attach_single_file_db системске процедуре.

6. Northwind база података

База података Northwind представља Мајкрософтову базу података која служи за демонстрирање рада њихових програма попут *SQL Server*-а и *Microsoft Access*- а. Она представља базу у коју се уносе подаци о продајним трансакцијама између имагинарне компаније "Northwind Traders" која се бави продајом хране и њених купаца и добављача.
На слици испод је дат дијаграм *Northwind* базе на коме су приказани сви ентитети са одговарајућим атрибутима као и везе између ентитета. [7]



Слика 6.1. *Northwind база*

7. Microsoft SQL Server

Microsoft SQL Server је програм за управљање релационим базама података који је развила компанија Мајкрософт. Примарна функција му је да служи као сервер за базе података односно да други програми могу преко њега да складиште и користе базе података било да су инсталирани на истом рачунару или другом рачунару на истој мрежи или интернету. Прва верзија програма је изашла 1989. године у сарадњи са фирмом "Sybase Corporation" под називом *SQL Server 1.0* за оперативни систем *OS/2*.

Када апликација поставља упит, модификује и додаје податке у *SQL Server* базу података, она користи *Structured Query Language* (SQL). *SQL* је стандардни језик који се користи за комуникацију са базама података. Такође користи екстензију *T- SQL* (*Transact-SQL*) која служи за рад са релационим базама података. *T- SQL* углавном додаје додатну синтаксу приликом писања процедура, и тиме утиче на подршку за трансакције. Све апликације које комуницирају са *SQL* Сервером комуницирају користећи *T- SQL*. [10]

Microsoft SQL Server такође користи за комуницикацију *TDS* протокол (*Tabular Data Stream Protocol*) служи за комуникацију између клијента и сервера као и апликација које користе *Linux* и *Unix*.

Microsoft SQL такође подржава *Open Database Connectivity*, односно скраћено *ODBC* технологију. *SQL* сервер од верзије 2005 поседује и подршку за *Web* сервисе, тј. за *Simple Object Access Protocol, Service Oriented Architecture Protocol* односно скраћено *SOAP W3C* стандард. То значи да клијенти који не користе *Windows* могу да комуницирају са *SQL Server-ом*. Од верзије *Microsoft SQL Server 2005* подржава и *Java Database Connectivity (JDBC) API* односно може да комуницира са Java апликацијама.

Постоје разне верзије *Microsoft SQL Server-a* са различитим опцијама намењени за различите групе корисника. Стандарде верзије су:

- Enterprise
- Standard
- Web
- Business Intelligence
- Workgroup
- Express

За поребе овог рада је коришћена верзија *Microsoft SQL Server 2008 R2*.

8. SQL Server Management Studio

SQL Server Management Studio (SSMS) је апикација преко које се врши конфигурација, управљање и администрација свих компоненти унутар *SQL Server-a*. Први пут се појавила у верзији *SQL Server 2005*, а њен претходник на серверу 2000 је апликација под називом *Enterprise Manager*. Главна функција апликације је да служи као графички интерфејс за рад са сервером. Централна карактеристика апликације је *Object Explorer* који омогућује претрагу, бирање и мењање објекта унутар сервера.

При покретању апликације прво је потребно повезати се на одоварајући сервер.

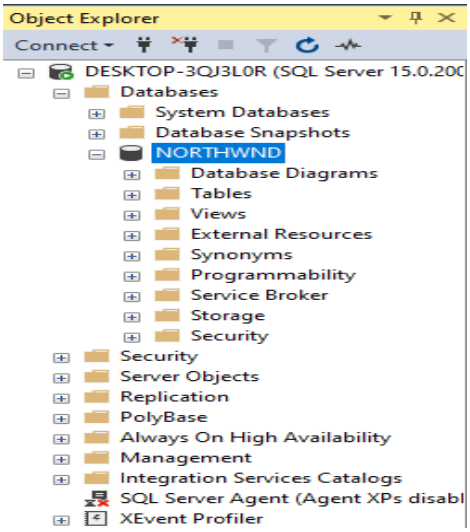
Слика 8.1 *Повезивање на сервер*

У прозору за повезивање ћемо оставити аутоматска подешавања, односно тип сервера ће бити *Database Engine*, име сервера (*local*), повезује се на локални сервер рачунара и аутентификација ће бити *Windows Authentication*. Потребно је кликнути на дугме "*Connect*".

9. Пример операција над базом података

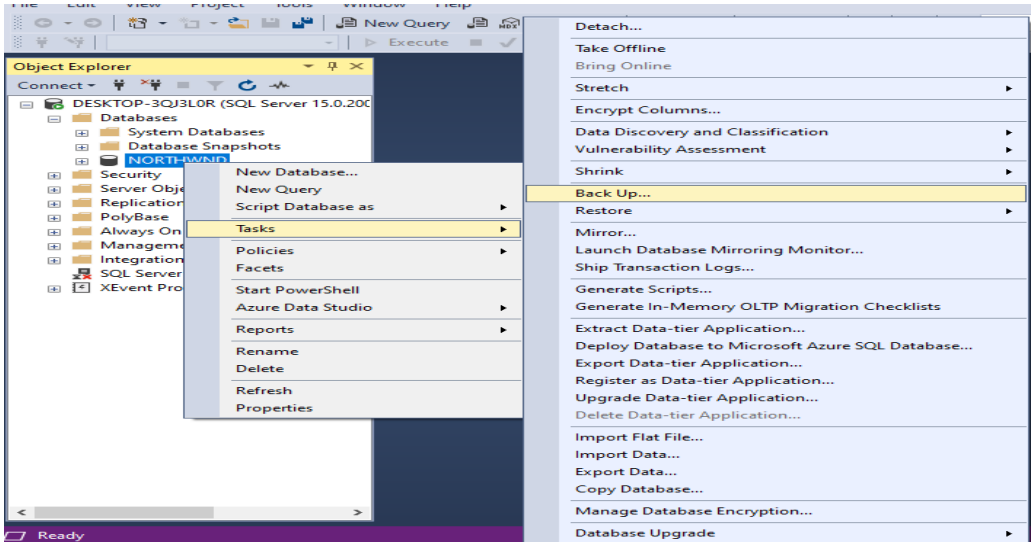
9.1 Прављење резервне копије базе података

За демонстрацију овог примера, коришћен је *Microsoft SQL Server* и његова екстензија *Microsoft SQL Server Management Studio*. Такође, као база података је коришћена *Northwind* база. Након што је покренут *SQL Server Management Studio* и извршена конекција са базом, у стаблу са леве стране екрана се може видети *Northwind* коју користимо, као што се види на слици испод.



Слика 9.1 *Стабло SQL Server-а*

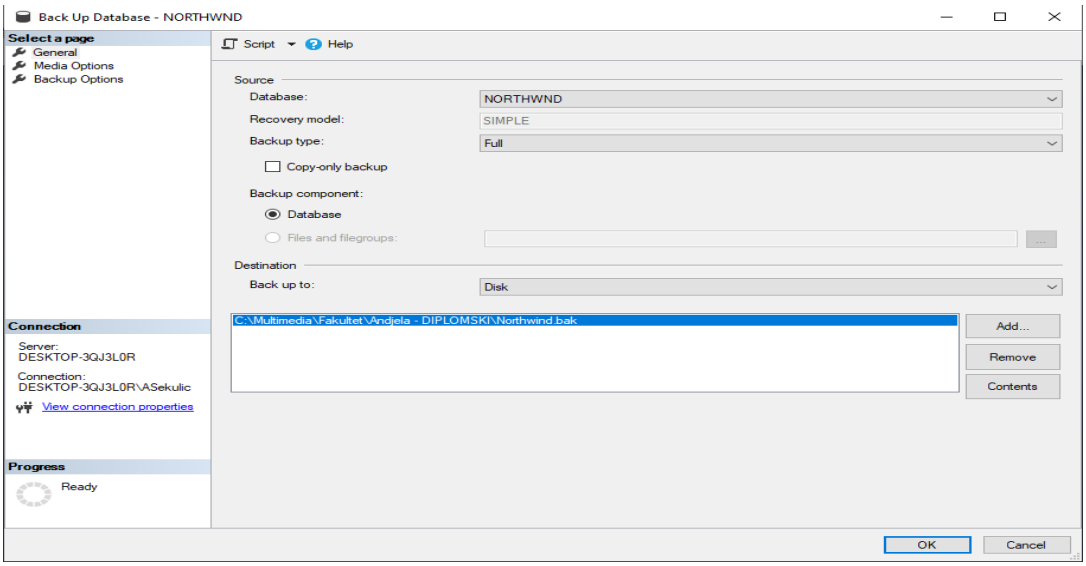
Десним кликом на базу *NORTHWIND* па избором опција *Tasks* → *Back Up* се приступа процесу прављења резервне копије саме базе. Приказ операција и опција над базом су приказане на слици испод.



Слика 9.2 – *Креирање резервне копије базе*

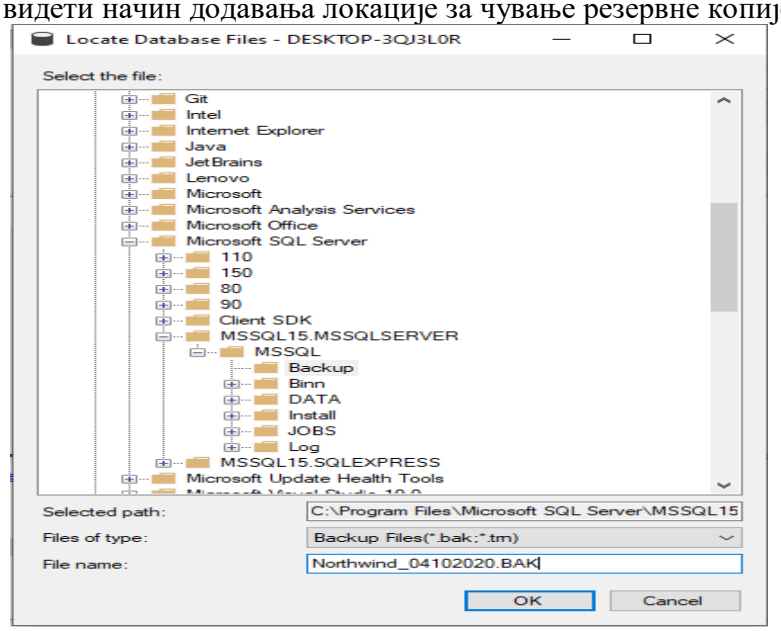
Након избора опције прављења резервне копије отвара се прозор који се састоји из три картице. У свакој од картица је потребно извршити одређена подешавања како би се успешно креирао жељени тип резервне копије.

Пре свега, потребно је из падајућег менија изабрати жељену базу података. Онда се бира жељени тип резервне копије (три типа су доступна у већини верзија *SQL Server*-а: *Full*, *Differential* или *Log*). У овом примеру је изабрано креирање потпуне резервне копије (*Full*). Након тога је потребно изабрати локацију чувања резервне копије. Потребно је избрисати предефинисану локацију, и кликом на дугме *Add...* отворити прозор у који је потребно унети локацију. Приказ прве картице је могуће видети на слици испод.



Слика 9.3 *Општа подешавања прављења резервне копије*

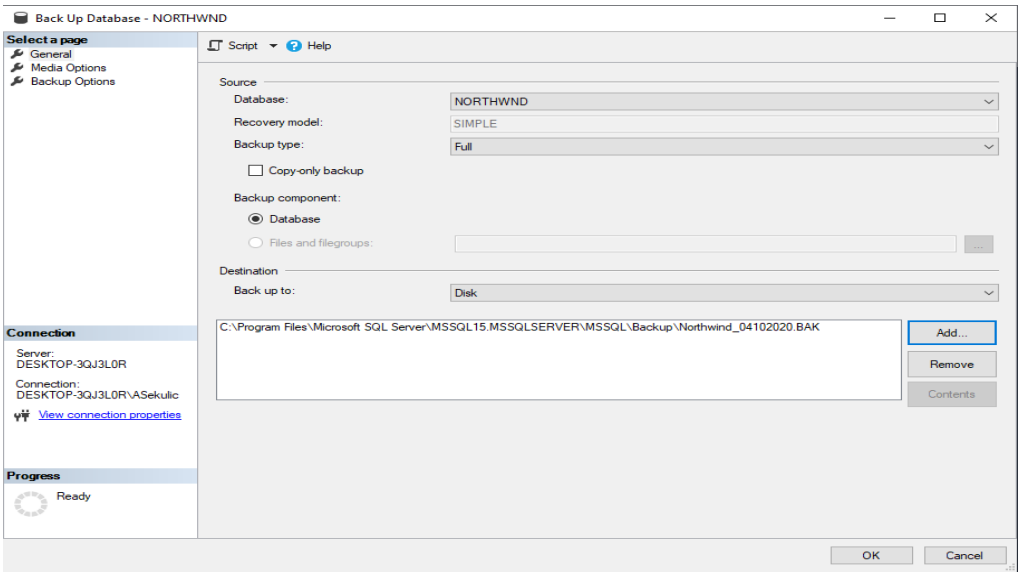
На слици испод је могуће видети начин додавања локације за чување резервне копије базе података.



Слика 9.4 *Локација резервне копије базе*

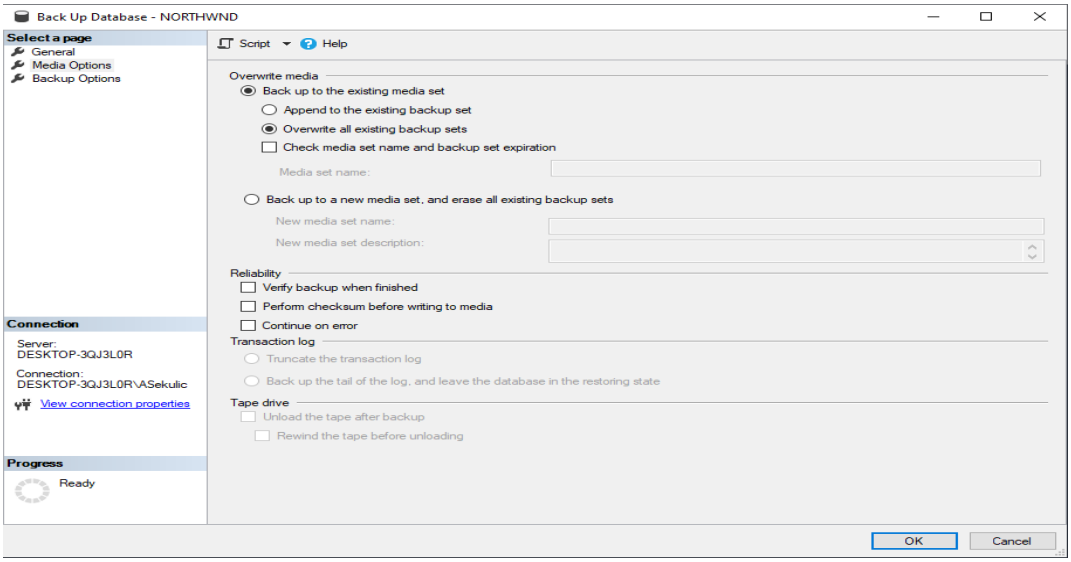
Битно је напоменути да је у овом примеру изабрана предефинисана локација за чување резервних копија. Такође, приликом додавања имена самог фајла неопходно је унети екстензију *.BAK* како би се резервна копија у будућности могла искористити и како би она била примењива. У случају именовања фајла без додавања поменуте екстензије, у будућности ће се фајлу у називу морати додати ова екстензија пре употребе.

Након избора локације и именовања датотеке, потврда се врши притиском на тастер ОК. Након тога се у пољу за локацију у првој картици може видети изабрана локација, као на слици испод.



Слика 9.5 *Преглед подешавања у првој картици*

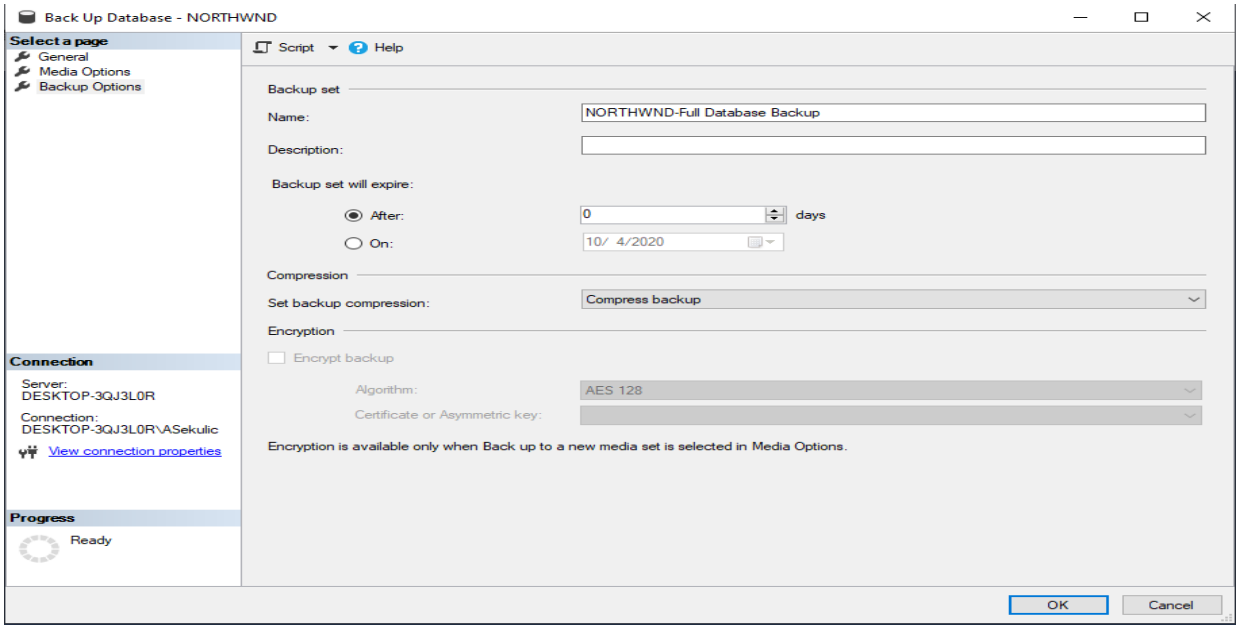
Следи наставак подешавања, пре креирања резервне копије базе. Избором опције *Media Options* се прелази у другу картицу, која изгледа као на слици испод.



Слика 9.6 *Подешавања у другој картици*

У другој картици је потребно изабрати опцију *Overwrite all existing backup sets* како би се при генерисању резервних копија прегазиле постојеће.

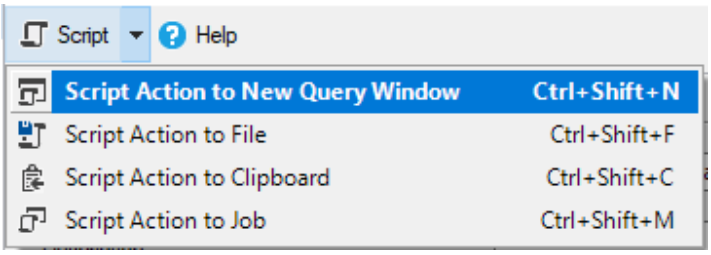
Избором опције *Backup Options* се врши прелаз у трећу картицу која изгледа као на слици испод.



Слика 9.7 *Подешавања у трећој картици*

Што се подешавања у трећој картици тиче, битно је напоменути опцију компресовања резервне копије. Ова опција је од изузетне важности поготово кад се ради о честим креирањима резервних копија база које су велике у смислу меморије коју заузимају. Избором опције креирања компресоване резервне копије, штеде се ресурси (време, меморија). Битно је напоменути да у неким старијим верзијама сервера није могуће компресовање резервних копија.

У овом кораку се при врху екрана може видети опција крирања скрипте (дугме *Script*). Кликом на стрелицу се отвара падајући мени који изгледа као на слици испод .



Слика 9.8 *Генерисање скрипте*

Притиском тастера на тастатури *Ctrl + Shift + N* или једноставним избором прве опције из падајућег менија се генерише скрипта за креирање резервне копије. Скрипта је приказана на слици испод.

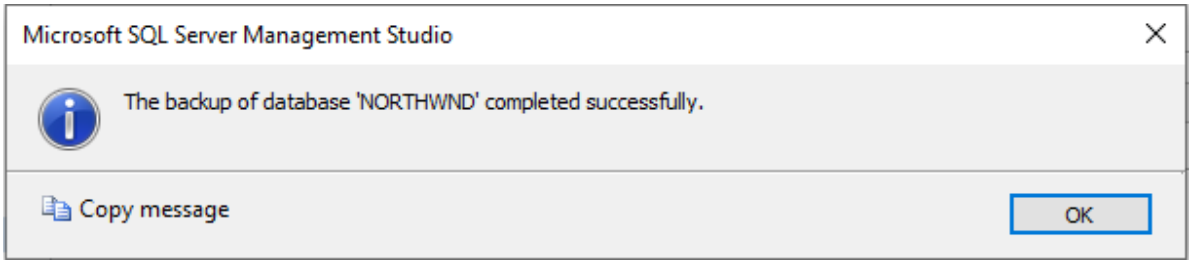
```
BACKUP DATABASE [NORTHWND] TO
DISK = N'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\Northwind_04102020.BAK'
WITH NOFORMAT, INIT,  NAME = N'NORTHWND-Full Database Backup',
SKIP, NOREWIND, NOUNLOAD, COMPRESSION,  STATS = 10
GO
```

Слика 9.9 *Генерисање скрипте*

Као што се може видети, скрипта описује операцију креирања резервне копије са свим изабраним опцијама. Приказан је назив базе, локација, назив резервне копије, објашњена опција компресије, као и још неколико опција.

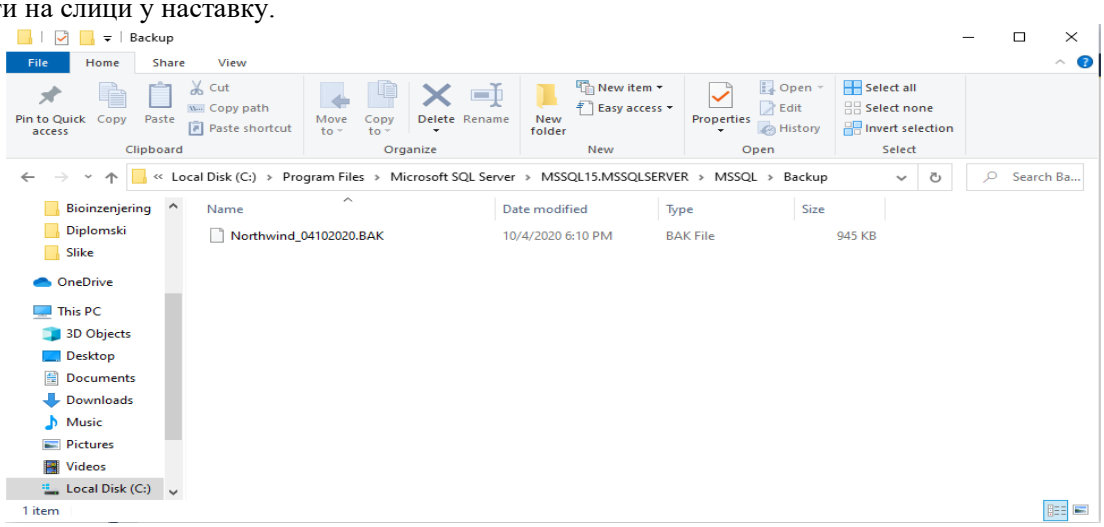
Након што извршена подешавања везана за прављење резервне копије базе података *Northwind*, у прозору се притиском на дугме ОК врши генерисање резервне копије. Потребно време је изузетно кратко, обзиром да је селектована опција компресије али и због мале величине изабране базе.

Након успешног креирања резервне копије, може се видети порука о успешном креирању, као што се види на слици испод.



Слика 9.10 *Генерисање скрипте*

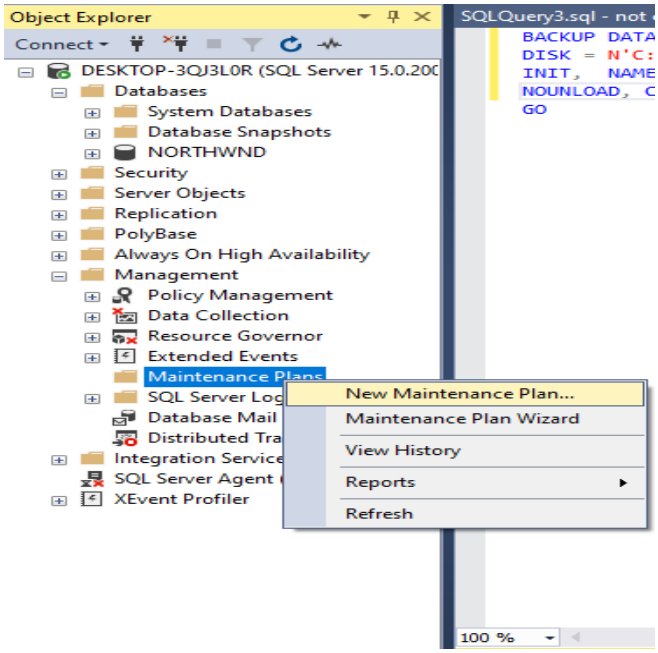
На крају, на изабраној локацији може видети креирана резервна копија (енг. *Back Up*). Приказ фајла се може видети на слици у наставку.



Слика 9.11 *Направљена резервна копија*

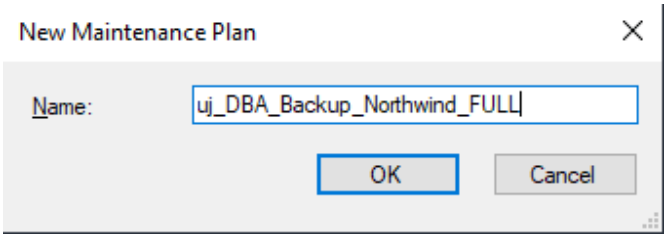
9.2 Креирање плана за генерисање резервних копија

Алтернатива ручном генерисању резервних копија базе података јесте креирање плана. На тај начин се све обавља аутоматизовано, тачно у одређено време. Отварањем фолдера *Management* из стабла са слике испод видимо фолдер *Maintenance Plans*. Десним кликом на тај фолдер па затим на опцију *New Maintenance Plan…* се приступа креирању новог плана одржавања резервних копија базе података.



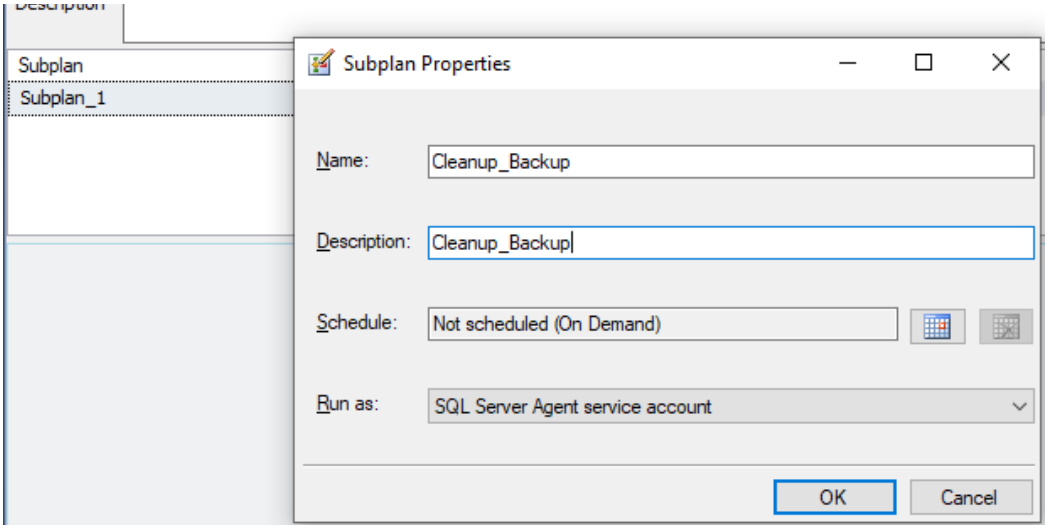
Слика 9.12 *Креирање плана одржавања резервних копија базе*

Потребно је именовати план одржавања пре свега, као што је приказано на слици испод.



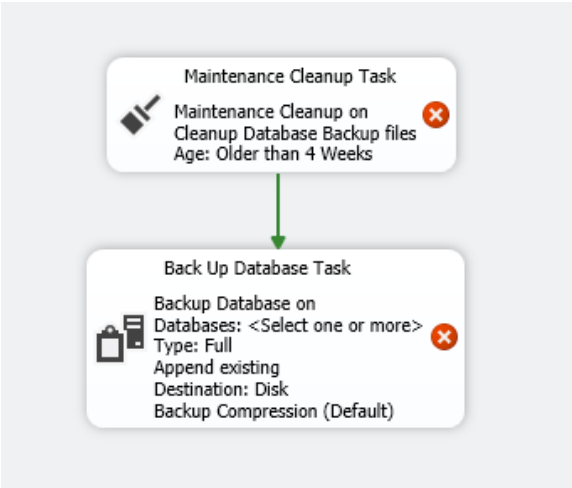
Слика 9.13 *Именовање плана одржавања*

Затим је потребно променити назив подлпана *Subplan_1* у *Cleanup_Backup*. Такође је и у поље за опис препоручљиво унети исти назив како би се касније лакше идентификовао подплан. На слици испод је приказан претходно описани поступак.



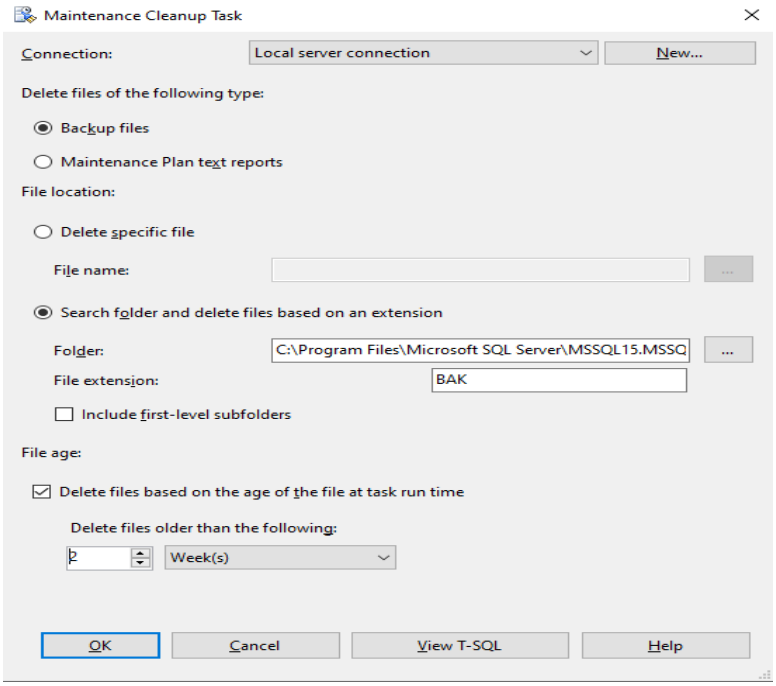
Слика 9.14 *Именовање под-плана*

Пошто је креиран и именован план одржавања, потребно је у пољу *Toolbox* пронаћи и у десни део екрана (испод подплана) превући *Maintenance Cleanup Task*, па одмах затим и *Back Up Database Task*. Након превлачења и позиционирања једног испод другог, то изгледа као на слици испод.



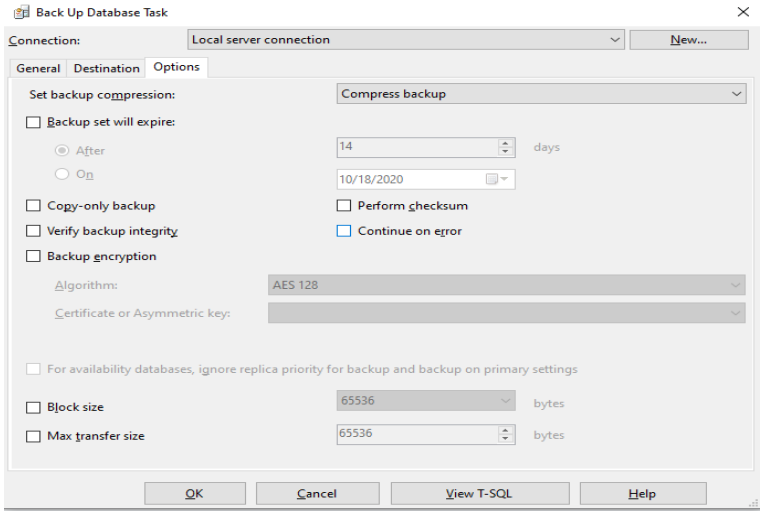
Слика 9.15 – *Задаци (енг. Maintenance Cleanup & Back Up Database Task)*

Дуплим кликом на задатке се врше подешавања. Што се тиче подешавања *Maintenance Cleanup Task*-а, подешавања су приказана на слици испод.



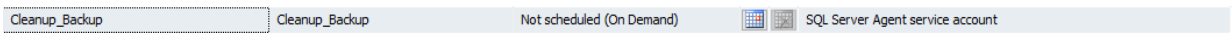
Слика 9.16 *Подешавања Maintenance Cleanup Task*-а

Следи приказ подешавања *Back Up Database Task*-а. Може се видети да је у три картице подешен овај задатак и да је у картици *Options* изабрано прављење резервне копије користећи технику компресовања.



Слика 9.17 *Подешавања Back Up Database Task*-а

Преостало је само још заказивање прављења резервне копије. То се ради кликом на календар који се може видети на слици испод. Након изабраног датума, успешно је креиран задатак који ће у тачно изабрано време вршити прављење резервне копије и, уколико је тако подешено, брисати старије верзије резервних копија.

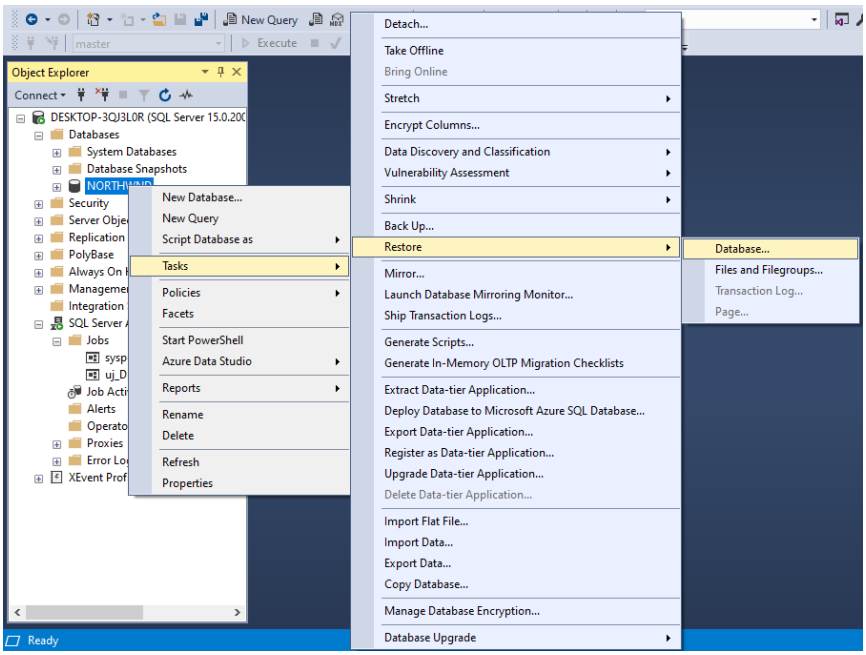


Слика 9.18 *Временско заказивање одржавања резервне копије базе података*

9.3 Обнављање резервне копије базе података

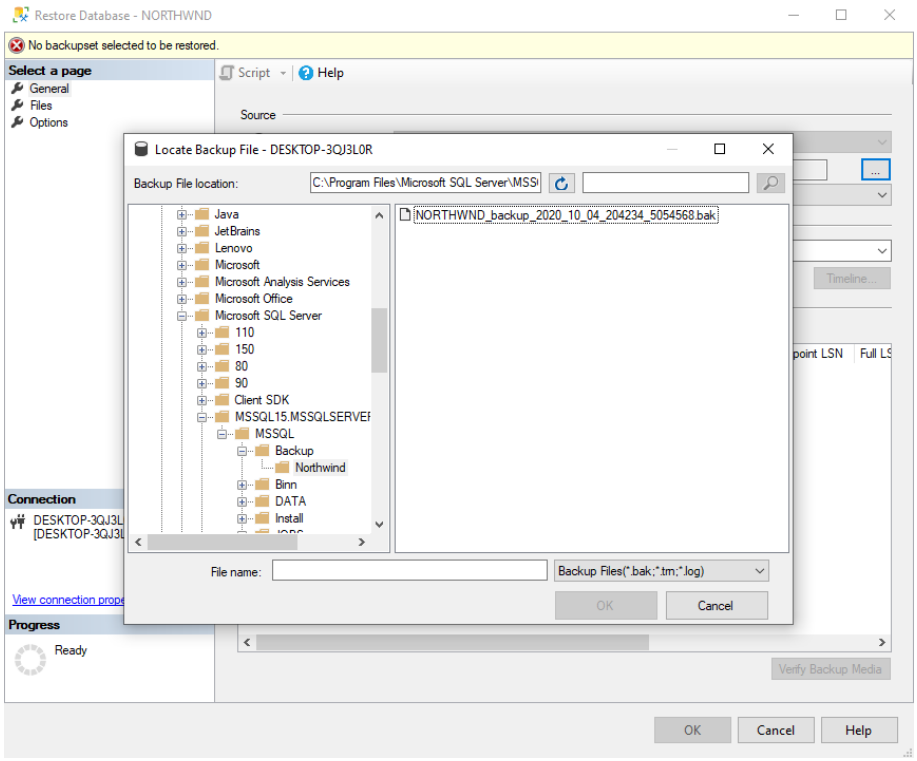
Обнављање резервене копије је једноставан поступак, с тим што се мора бити веома опрезан. Превид или грешка у појединим корацима овог поступка може имати неопозиве негативне последице на базу података. На пример, може се десити трајно губљење података.

За обнављање резервне копије, потребно је десним кликом изабрати базу података, па затим *Tasks* → *Restore* → *Database*. Приказ опција је приказан на слици испод.



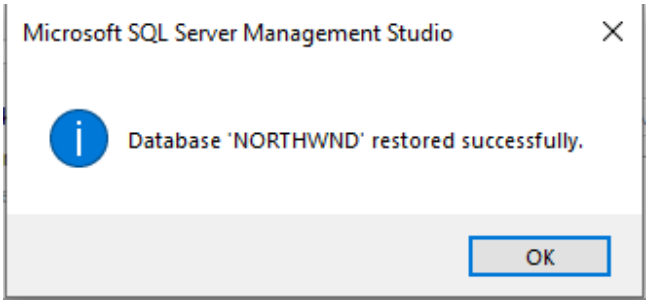
Слика 9.19 *Обнављање базе података*

Избором *Device* као извора, врши се избор локације резервне копије коју је потребно обновити. У овом примеру је коришћена ручно генерисана резервна копија, десним кликом на *uj_DBA_Backup_Northwind_FULL.Cleanup_Backup* па затим *Start Job at Step*, као што се види на слици испод.



Слика 10.20 *Избор путање до резервне копије базе*

Након изабране базе односно резервне копије базе који желимо да обновимо, избор је потребно потврдити кликом на дугме OK. На слици испод се може видети порука о успешној обнови базе.



Слика 10.21 *Успешна обнова базе података*

10. Закључак

У данашње време архивирање и доступност података заузима јако битно место у развоју пословања. У свим сферама пословног света врши се процес дигитализације и самим тим је неопходно чување података, као и њихова доступност ради уноса, измене и ажурирања тих података.

Најгоре што у пословању може да се деси јесте губљење података небитно из ког разлога. Зато се треба осигурати и правити често резервне копије и чувати их на различитим медијима. Иако не може гарантовати потпуну заштиту, онда може свести губљење података на минимум.

Такође је у раду приказано да *SQL Server* има подршку за релативно лако прављење резервних копија базе података, као и коришћење тих копија за поновно успостављање базе података.

11. Литература

[1] Microsoft Corporation: Administering a Microsoft SQL Server 2000 Database, Delivery Guide, 2000,

[2] Бранислав Лазаревић, Зоран Марјановић, Ненад Аничић, Срђан Бабарогић: Базе података, Факултет организационих наука, Београд, 2018

[3] Младен Веиновић, Горан Шимић, Александар Јевремовић, Милан Таир: Базе података, Универзитет Сингидунум, Београд, 2018

[4] Павлетић-Лажетић Г. : Основе релационих база података, Математички факултет, Београд 2000.

[5] Ross Mistry, Stacia Misner: Introducing Microsoft SQL Server 2008, Microsoft Press, Washington, 2010.

[6] Michael Lee, Gentry Bieker:Mastering SQL Server 2008., Wiley Publishing, Inc.Indianopolis, Indiana, 2009.

Веб сајтови:

[7] http://www.link-university.com/lekcija/Podr%C5%A1ka-za-SQL-Server-baze-podataka/5277 приступљено септембар 2020.

[8] https://www.vpsle.edu.rs/wp-content/uploads/2016/04/INFORMACIONI-SISTEMI-SKRIPTA-2016.pdf, Информациони системи, Висока пословна школа струковних студија у Лесковцу, скрипта, приступљено септембар 2020.

[9] https://cdn.technologyadvice.com/wp-content/uploads/2017/05/Fotolia_130308566_Subscription_Monthly_M-700x408.jpg приступљено септембар 2020.

[10] https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql-compatibility-level?view=sql-server-ver15 приступљено септембар 2020.