KAPITOLA 6

Zabezpečení serveru a údajů

V této kapitole:

- Správa uživatelských práv
- Transparentní šifrování
- Auditování

Současný propojený svět přináší obrovské příležitosti nejen pro byznys, ale prakticky pro všechny oblasti života. Cenově dostupné, výkonné počítače, vysokorychlostní připojení, Internet a rozmach mobilních zařízení společně vytváří globální síť dříve netušených možností. Tato globální síť ale také přináší nežádoucí problémy. Vzájemně propojená elektronická zařízení jsou ohrožována útoky.

Firmy a organizace musí zajistit nejen rozvoj a pokrok, ale i dosažení odpovídající úrovně bezpečnosti. Zabezpečení nesmí omezovat růst a služby zákazníkům, musí podporovat nové možnosti, jako je například elektronická spolupráce partnerů nebo elektronické obchodování se zákazníky. Zároveň musí minimalizovat rizika neoprávněného přístupu k citlivému duševnímu vlastnictví nebo odcizení identity. V maximální možné míře je třeba omezovat ztráty produktivity zaměstnanců.

Přiměřené zabezpečení informací a informačních systémů ve firmách a organizacích předepisuje i mezinárodní legislativa, konkrétně norma ISO/IEC 17799:2000. Tato norma definuje informaci jako aktivum organizace, které, tak jako jiná významná obchodní aktiva, má zhodnotit organizaci, a proto musí být informace přiměřeně chráněné.

Pojem bezpečnost databáze může být vyložen dvěma způsoby. Na jedné straně je bezpečnost chápána jako zabezpečení údajů před možnou ztrátou, na straně druhé je bezpečnost jako ochrana pře možným zneužitím údajů, případně před průnikem nepovolaných osob (hackerů) do aplikace databáze.

Oba zmíněné faktory spolu úzce souvisejí. Například z důvodu možné ztráty údajů je rozumné pravidelně zálohovat všechny údaje z databáze na nějaké médium. Takto ale vzniká riziko, že při nedostatečném technickém a organizačním zabezpečení je možné údaje odcizit a případně zneužít.

V úvodu kapitoly věnované zabezpečení databázového serveru zdůrazníme jednu důležitou a osvědčenou zásadu, a tou je paranoia. Nebojte se, pořád čtete knihu věnovanou SQL Serveru 2008, ne zatoulané stránky z psychologické nebo psychiatrické literatury.

V takové literatuře se popisuje, jak paranoidní stav léčit, zatímco v následujícím textu budeme naopak propagovat způsob, jak jistou míru paranoii vyvolat. Při administraci databáze a nastavování přístupových práv nevěřte nikomu a uživatelům už vůbec ne.

Nemůžete spoléhat na to, že se uživatelé budou chovat zodpovědně, a už vůbec ne na to, že se budou chovat korektně. Vytvořit databázovou aplikaci nad účtem systémového administrátora a zveřejnit pro všechny klienty jeho přístupové údaje určitě nezajistí bezpečný provoz databáze. Když už jsme u psychologie, zmíníme i problém s délkou hesla.

Pokud je heslo jednoduché a lehko zapamatovatelné, dá se i snadno odhalit. Pokud je naopak pro heslo použitá 25znaková kombinace malých a velkých písmen a číslic, pravděpodobně si každý heslo někam napíše, a tak stačí prozkoumat žluté papírky nalepené na stole, případně nápisy tužkou na okraji monitoru, nebo na spodní straně klávesnice.

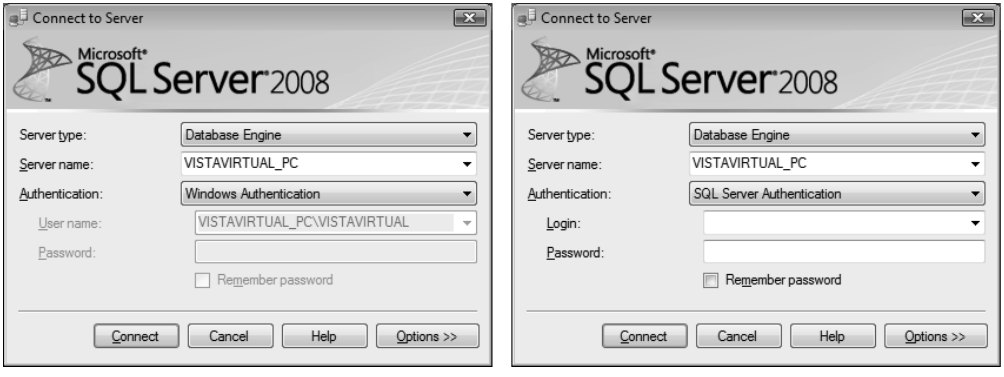
Správa uživatelských práv

Databáze je jedním z typických příkladů víceuživatelského prostředí. K údajům v databázi přistupuje více uživatelů, a to různým způsobem. Někteří uživatelé mohou údaje do databáze zapisovat, případně je mazat, jiní mají povolené jen čtení údajů. Proto je velmi důležité definovat přístupová práva jednotlivých uživatelů k databázovým objektům a specifikovat rozsah jejich oprávnění. Uživatelský přístup má na starosti správce databáze, většinou ve spolupráci se správcem systému.

Poznámka: Je možné, že jako uživatelé databáze, analytici, případně vývojáři databázových aplikací nepřijdete s administrací databáze vůbec do styku. Ale je možné, že vám bude svěřený nějaký logický úsek databázové aplikace, kde prostě budete muset přístupová práva uživatelů nastavovat, nebo měnit.

SQL Server 2008 je na rozdíl od konkurenčních produktů, které fungují na různých operačních systémech (UNIX, LINUX, Windows, mainframe atd.), jednoplatformový databázový server. Jako takový je úzce svázaný s operačním systémem Windows, a to bez ohledu na verzi operačního systému, ať už jde o serverové operační systémy Windows Server 2008, Windows Server 2003, nebo klientské systémy Windows 7, Vista, Windows XP. Pro autentifikaci a autorizaci je možné využít dva způsoby zabezpečení:

- Integrované zabezpečení Windows
- Autentifikace SQL Serveru



Obrázek 6.1 Přihlašování pomocí Windows autentifikace (vlevo) a SQL Server autentifikace (vpravo)

Windows autentifikace využívá bezpečnostní vlastnosti operačních systémů Windows Server a umožňuje SQL Serveru sdílet uživatelská jména a hesla používaná ve Windows. Uživatel si potom nemusí pamatovat různá hesla a uživatelská jména pro operační systém a SQL Server. Stačí mu měnit hesla v operačním systému.

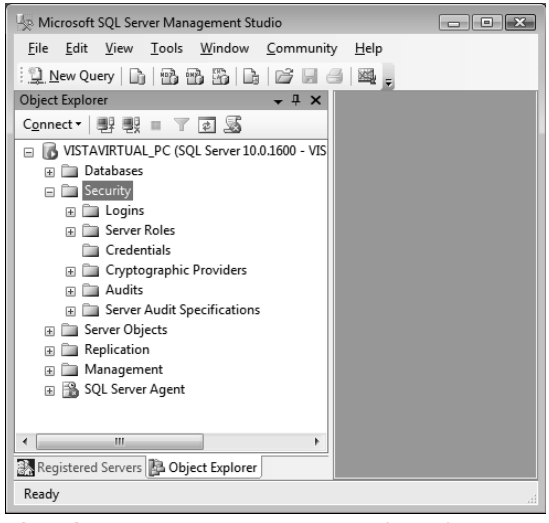
Windows autentifikace v SQL Serveru funguje tak, že když uživatel přistupuje k SQL Serveru, tento získá informace o uživateli z operačního systému. Pokud má tento uživatel povolený přístup k SQL Serveru, je automaticky na databázový server připojený na SQL

Server. V módu autentifikace SQL Serveru je možné k serveru připojit i klienty, případně aplikace, kteří nemají přístupové účty na úrovni operačního systému.

Po úspěšném přihlášení na SQL Server, ať už za pomoci Windows nebo SQL autentifikace, databázový server verifikuje, zda je uživatel platným uživatelem pro databázi, ke které chce přistupovat. Tento proces se nazývá autorizace.

Správa uživatelských práv v prostředí SQL Server Management Studio

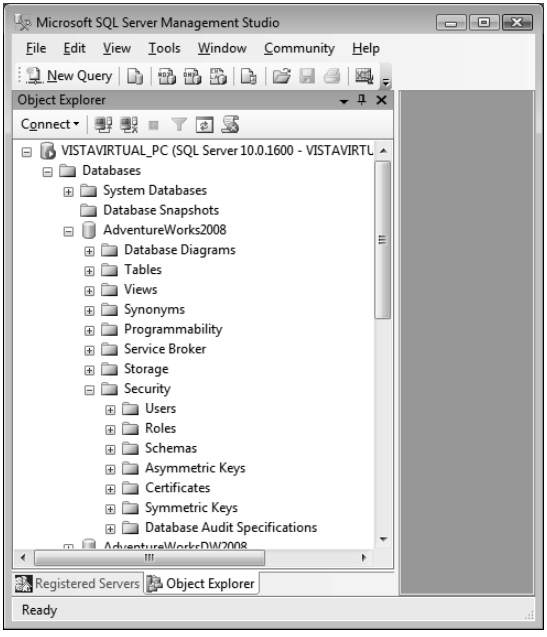
Administrátorské úkony lze vykonávat buď v grafickém uživatelském prostředí aplikace SQL Server Management Studio, nebo zadáváním příkazů v jazyku SQL pomocí klientské konzolové aplikace. SQL Server má na vytváření a manipulaci s uživatelskými účty implementované uložené procedury.



Obrázek 6.2 SQL Server Management Studio – zabezpečení na úrovni databázového serveru

Většina objektů týkajících se zabezpečení, tedy uživatelské účty, role, objekty pro správu šifrování, se na platformě SQL Server 2008 vyskytují ve dvou úrovních. Vážou se buď k serveru, nebo k databázi, přičemž jsou ještě provázané i navenek. Protože jsou databáze pod správou serveru, některé objekty od něj dědí. Proto je v SQL Server Management Studiu záložka **Security** na dvou úrovních. Jednak pro databázový server, jednak samostatně pro jednotlivé databáze.

Upozornění: Pro přidělování přístupových práv musíte mít vy sami, tedy přesněji váš uživatelský účet, dostatečné oprávnění. To znamená, že musíte být členem role sysadmin nebo securityadmin.



Obrázek 6.3 SQL Server Management Studio – Zabezpečení na úrovni databáze

Vytvoření nového uživatelského účtu na serveru

Vytvoření nového uživatele bylo stručně zmíněné už ve čtvrté kapitole. Pro práci s databází je potřeba mít účet navíc přidělený. Nového uživatele, nebo přesněji řečeno nový uživatelský účet, lze vytvořit pomocí vizuálního nástroje SQL Server Management Studio nebo prostřednictvím konzolové aplikace příkazem CREATE LOGIN. Zjednodušená syntaxe příkazu CREATE LOGIN je:

```
CREATE LOGIN jmeno WITH PASSWORD='heslo' [MUST_CHANGE],
    | DEFAULT_DATABASE = database
    | DEFAULT_LANGUAGE = language
    | CHECK_EXPIRATION = { ON | OFF}
    | CHECK_POLICY = { ON | OFF}
    | CREDENTIAL = credential_name
```

Parametry přihlašovací jméno a heslo není potřeba nijak zvlášť komentovat. Pozornost si však zaslouží parametr databáze. Určuje databázi, do které se uživatel může přihlásit. Kdybyste tento parametr nezadali, mohl by se uživatel přihlásit do databáze MASTER, což z bezpečnostního hlediska málokdy vyhovuje. Proto doporučujeme tento parametr vždy zadat.

Upozornění: Je potřeba rozlišovat mezi vytvořením uživatelského účtu na úrovni serveru pomocí příkazu CREATE LOGIN a vytvořením uživatele pro databázi příkazem CREATE USER, který bude popsaný dále.

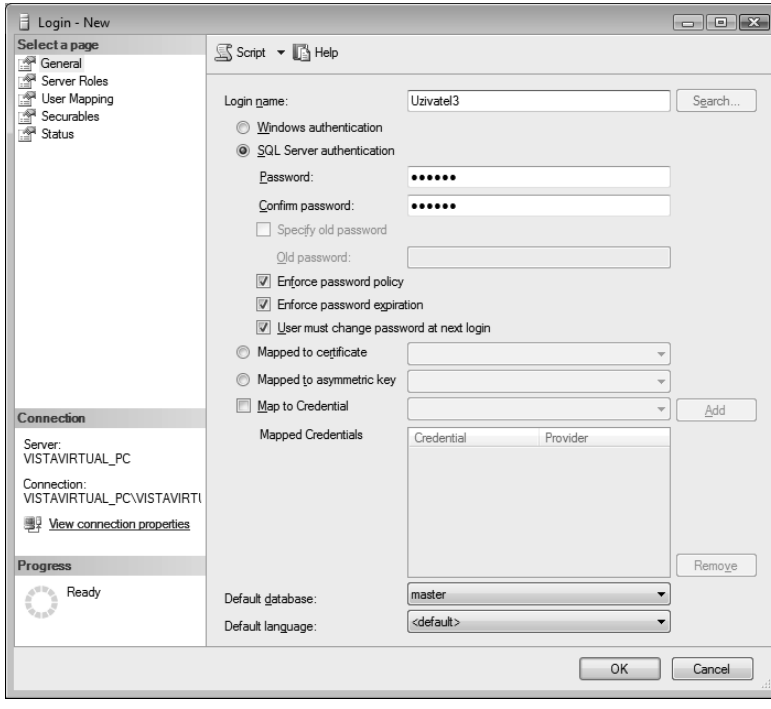
Pro vytvoření nového uživatele, například se jménem JanNovak, použijte příkaz:

```
CREATE LOGIN [JanNovak] WITH PASSWORD = 'honza', DEFAULT_DATABASE=[Test],
```

Pomocí takto zjednodušeného příkazu je definováno uživatelské jméno, počáteční heslo a databáze. Z bezpečnostních důvodů se doporučuje nechat heslo expirovat a donutit tak uživatele, aby si ho při prvním použití změnil.

```
CREATE LOGIN [JanNovak] WITH PASSWORD='honza' MUST_CHANGE,
DEFAULT_DATABASE=[Test], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
```

V prostředí SQL Server Management Studio lze vytvořit nového uživatele pomocí místní nabídky **New Login** ve složce **Security**. Postupně se na jednotlivých podstránkách dialogu definují přihlašovací parametry, role a ostatní parametry. Všimněte si, že dialog obsahuje jen záložku pro definování serverových rolí. Role pro jednotlivé databáze se definují ve složkách **Security** pro jednotlivé databáze.



Obrázek 6.4 Vytvoření uživatelského účtu pomocí dialogu v nástroji SQL Server Management Studio

Na základě vizuálního návrhu bude vygenerovaný skript:

```
USE [master]
GO
CREATE LOGIN [Uzivatel3] WITH PASSWORD=N'katmai' MUST_CHANGE,
DEFAULT_DATABASE=[master], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO
```

Parametry serverového uživatelského účtu lze měnit pomocí příkazu ALTER LOGIN, například:

```
ALTER LOGIN [Uzivatel1] WITH PASSWORD=N'katmai1'
```

Pro odstranění serverového uživatelského účtu je určený příkaz DROP LOGIN:

```
DROP LOGIN jmeno
```

Správa uživatelů pomocí systémových uložených procedur

Možnost správy uživatelských účtů pomocí systémových uložených procedur bude pravděpodobně v budoucích verzích SQL Serveru vypuštěná nebo zůstane pouze z důvodu zachování kompatibility. Jednoznačně doporučujeme používat příkazy uvedené výše. Pro vytvoření nového účtu je určená uložená procedura SP_ADDLOGIN. Její parametry jsou:

```
jméno serverového uživatelského účtu    [ , [ @passwd = ] 'heslo' ]
    [ , [ @defdb = ] 'databáze' ]
    [ , [ @deflanguage = ] 'jazyk' ]
    [ , [ @sid = ] sid ]
    [ , [ @encryptopt = ] 'šifrovací_parametry' ]
```

Příklad vytvoření uživatele pomocí uložené procedury:

```
EXEC sp_addlogin 'JanNovak', 'honza', 'Test'
```

Heslo uživatele je možné změnit i pomocí procedury SP_PASSWORD:

```
sp_password [ [ @old = ] 'stare_heslo' , ]
    { [ @new =] 'nove_heslo' }
    [ , [ @loginame = ] 'jmeno' ]
```

Pokud není zadaný parametr jméno, změní se heslo aktuálně přihlášeného uživatele. Pro uživatele JanNovak bude heslo změněno příkazem:

```
EXEC sp_password 'honza', 'jenda'
```

Pro odebrání uživatele je určená uložená procedura SP_DROPLOGIN:

```
sp_droplogin [ @loginame = ] 'uživatelské jméno'
```

Například:

```
EXEC sp_droplogin 'JanNovak'
```

Pro nastavení práv uživatele pro přístup k databázi je určená uložená procedura SP_GRANTDBACCESS:

```
sp_grantdbaccess [@loginame =] 'jméno'
    [,[@name_in_db =] 'jméno_v_databázi' [OUTPUT]]
```

Uživateli JanNovak bude umožněn přístup k aktuální databázi příkazem:

```
EXEC sp_grantdbaccess ' LLHOME\Administrator', 'JanNovak'
```

Vytvoření uživatele v databázi

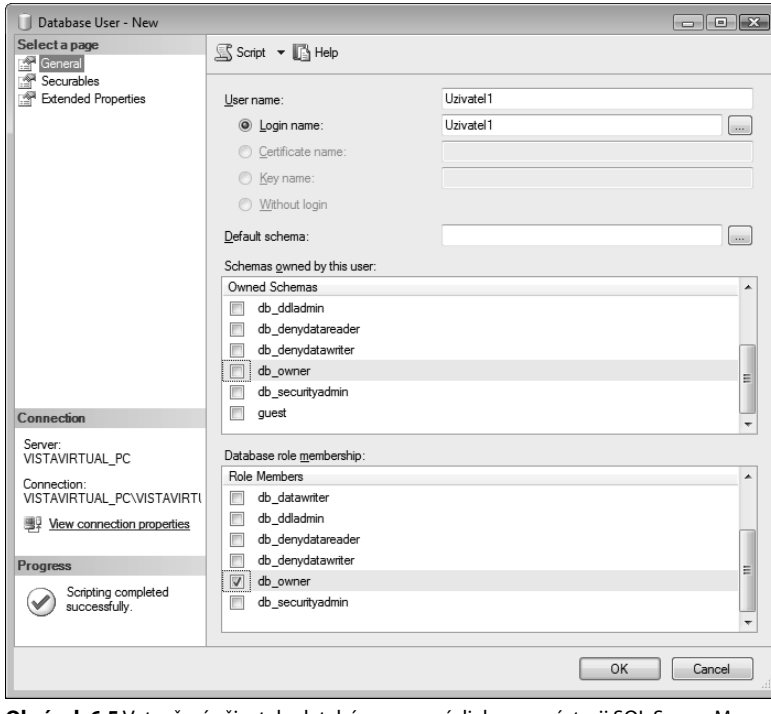
Pro vytvoření uživatele na úrovni databáze slouží příkaz CREATE USER. Tento příkaz může zavolat jen správce databáze nebo uživatel, který má k tomu delegovaná příslušná privilegia. Základní syntaktický předpis toho příkazu je:

```
CREATE USER user
   [{ FOR | FROM } LOGIN login_name] [WITH DEFAULT_SCHEMA = schema]
```

Například:

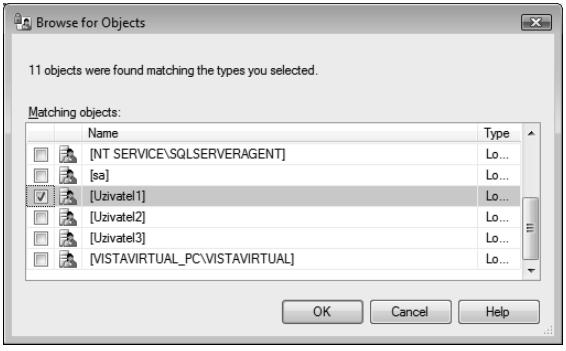
```
CREATE USER [Uzivatel1] FOR LOGIN [Uzivatel1]
```

Pro vytvoření nového uživatele pro příslušnou databázi je k dispozici i návrhový dialog v SQL Server Management Studiu. Aktivuje se prostřednictvím místní nabídky pomocí položky **New User** ve složce **Security** příslušné databáze.



Obrázek 6.5 Vytvoření uživatele databáze pomocí dialogu v nástroji SQL Server Management Studio

Uživatelský účet pro jednotlivé databáze může být vytvořen pouze tehdy, existuje-li stejný účet pro databázový server, tedy stejný účet byl dříve vytvořen pomocí příkazu CREATE LOGIN. Při vytváření účtu je možné vybrat účet z nabídnutého dialogu.



Obrázek 6.6 Dialog pro výběr uživatelského účtu za účelem vytvoření uživatele databáze

Pokus o vytvoření uživatele, který nemá vytvořený login, vede k chybě:

```
CREATE USER NovakAlfons

Msg 15007, Level 16, State 1, Line 1
'NovakAlfons' is not a valid login or you do not have permission.
```

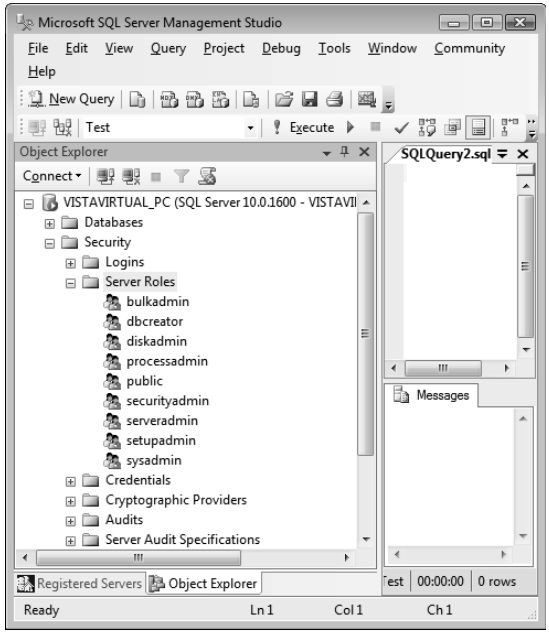
Poznámka: Rozdělení správy na dvě úrovně, tedy úroveň serveru a databáze, podporuje i teorie databází. Podle ní je správce (administrátor) systému řízení báze dat osoba zodpovědná za činnost a používání tohoto sytému, kdežto správce (administrátor) báze dat (Data Base Administrator) je osoba zodpovědná za návrh, vývoj, zabezpečení, údržbu a používání příslušné databáze.

Při pohledu na dialog pro vytvoření nového uživatele jste si určitě všimli, že do hry vstupují dva faktory – role a schémata.

Role

Jakmile začnete vytvářet uživatele pro informační systém, i když třeba jen na úrovni malé nebo střední firmy, a začnete jim přidělovat přístupová práva, zjistíte, že pro mnoho uživatelů nastavujete stejná práva. Ukazuje se, že bude vhodné vytvářet skupiny uživatelů se stejnými přístupovými právy, které prakticky kopírují funkční hierarchii firmy, ve které pracuje více zaměstnanců ve stejném pracovním zařazení. Proto jsou důležitou součástí přístupových práv role. Role umožňují sdružovat uživatele do skupin. Mezi uživateli a rolemi je vztah M:N, to znamená, že je nejen více uživatelů přiřazených k jedné roli, ale i opačně, jeden uživatel může mít přístup k více rolím.

Role se na platformě SQL Server 2008 dělí na serverové a databázové. Serverové role se týkají operací a úkonů na úrovni serveru. Jestliže někoho umístíme do určité role, může tento uživatel provádět operace povolené pro tuto roli. Serverové role jsou předdefinované, nejsou databázově specifické a nemohou být upravovány. Jejich seznam si můžete prohlédnout v SQL Server Management Studiu ve složce **Security** a podsložce **Server Roles**.



Obrázek 6.7 Seznam serverových rolí

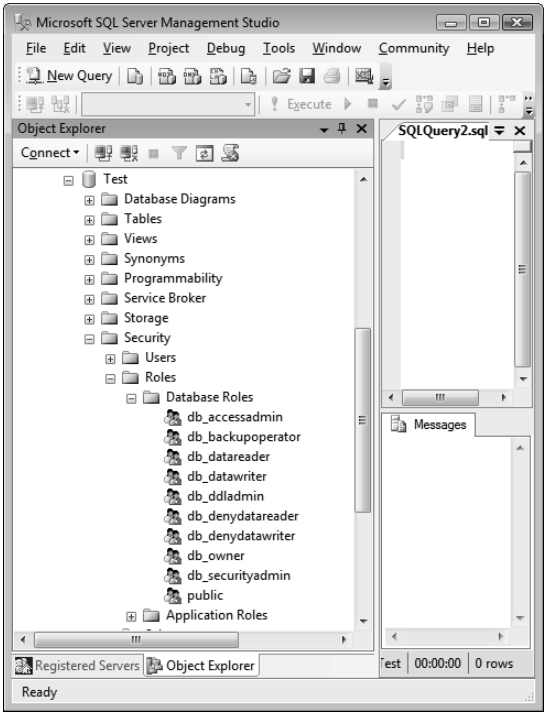
Databázové role jsou specifické pro danou databázi a umožňují přístup jednotlivému uživateli nebo skupině uživatelů k dané databázi v rozsahu, který je určený tou kterou rolí. Existují dva typy rolí – předdefinované a uživatelem definované role. Předdefinované databázové role jsou standardem v SQL Serveru. Každá databáze v SQL Serveru tyto role obsahuje a nelze je z databáze odebrat. Jsou databázově specifické a nemohou být upravované. Seznam předdefinovaných databázových rolí si můžete prohlédnout v nástroji SQL Server Management Studio ve složce Security pro příslušnou databázi, v podsložce Roles.

Uživatelem definované databázové role se rozdělují na standardní a aplikační. Toto rozdělení podporuje i SQL Server Management Studio, ve kterém jsou v okně **Object Explorer** ve struktuře příslušné databáze dvě podsložky:

- **Database Roles**
- **Application Roles**

Aplikační role neumožňuje přiřazovat uživatele. Proto takovou roli s určenými přístupovými právy může aktivovat kterýkoliv uživatel nezávisle na jeho přístupových právech v databázi. Okamžitě když uživatel aktivuje aplikační roli, zapomene na přístupová práva určená standardní rolí a využívá přístupová práva určená aplikační rolí.

Zajímavá je databázová role public, protože jejím členem je každý nově vytvořený uživatel databáze.



Obrázek 6.8 Seznam databázových rolí

Poznámka: Je třeba správně vnímat rozdíl mezi rolí public a rolí datareader, která má oprávnění jen pro čtení údajů. Role public je společná pro všechny uživatele a je možné využít ji pro správu bezpečnosti. Udělením nebo zamítnutím přístupových práv pro tuto roli budou ovlivněni všichni uživatelé příslušné databáze.

V databázích lze vytvořit role s určenými přístupovými právy pomocí příkazu CREATE ROLE. Zjednodušeně můžeme definovat syntaxi tohoto příkazu následovně:

```
CREATE ROLE jmeno_role;
```

Role je možné vytvorit také voláním systémové uložené procedury SP_ADDROLE:

```
sp_addrole [ @rolename = ] 'role'
    [ , [ @ownername = ] 'vlastnik' ]
```

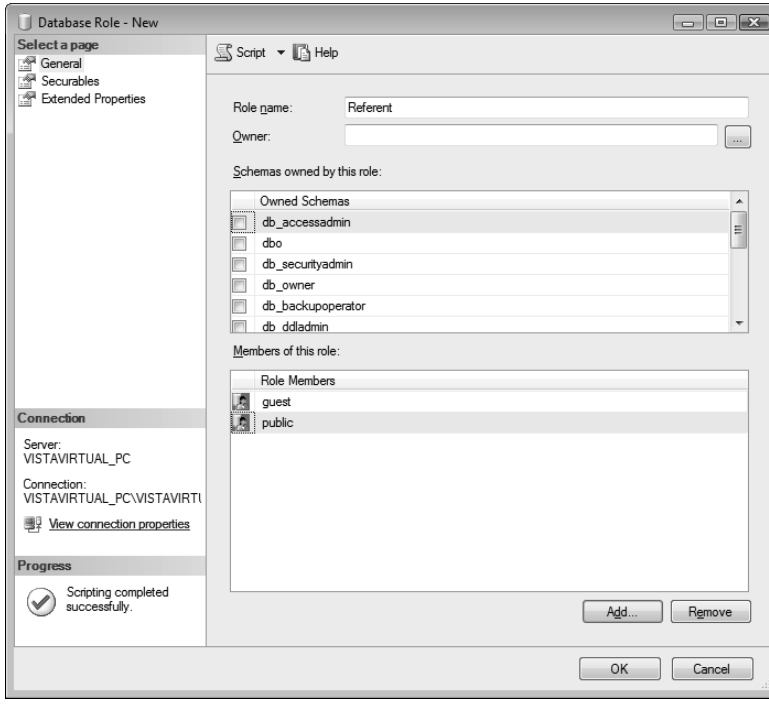
Tato možnost je ponechána jen z důvodu kompatibility. V nové verzi se její použití nedoporučuje.

Například:

```
CREATE ROLE Analytik
```

Aby mělo vytvoření role praktický význam, je třeba roli přidělit nějaká privilegia pomocí příkazu GRANT, například:

```
GRANT create table, create view  TO Analytik;
```



Obrázek 6.9 Dialog pro vytvoření nové databázové role v SQL Server Management Studiu

Po definování jednotlivých rolí do nich můžete přidat uživatele pomocí volání systémové procedury SP_ADDROLEMEMBER:

```
sp_addrolemember [ @rolename = ] 'role' ,
    [ @membername = ] 'nazev_uctu'
```

Například uživatele JanNovak zařadí administrátor mezi analytiku příkazem:

```
EXEC sp_addrolemember 'Analytik', 'JanNovak'
```

Pro různé účely je potřeba reportování aktuálního stavu zabezpečení, tedy seznam uživatelů, rolí a podobně. Pro výpis seznamu uživatelů je určená uložená procedura SP_HELPUSER:

```
sp_helpuser [ [ @name_in_db = ] 'security_account' ]
```

Například:

```
EXEC sp_helpuser
```

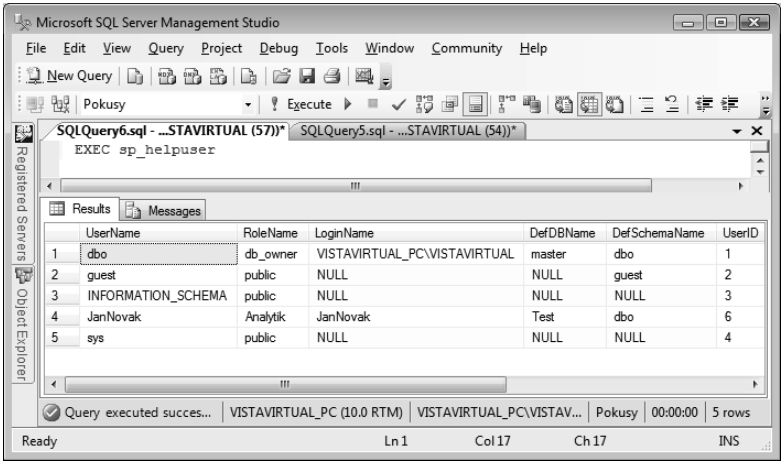
Údaje budou vypsané ve tvaru:

Informace o rolích získáte aktivací uložené procedury SP_HELPROLE.

```
sp_helprole [ [ @rolename = ] 'role' ]
```

Například:

```
EXEC sp_helprole
```



Obrázek 6.10 Výpis informací o uživatelích

Údaje získáte v tvaru:

```
RoleName
------------------------------
public
Analytik
db_owner
db_accessadmin
db_securityadmin
db_ddladmin
db_backupoperator
db_datareader
db_datawriter
db_denydatareader
db_denydatawriter
```

Komplexní údaje získáte spuštěním uložené procedury:

```
sp_helplogins [ [ @LoginNamePattern = ] 'login' ]
```

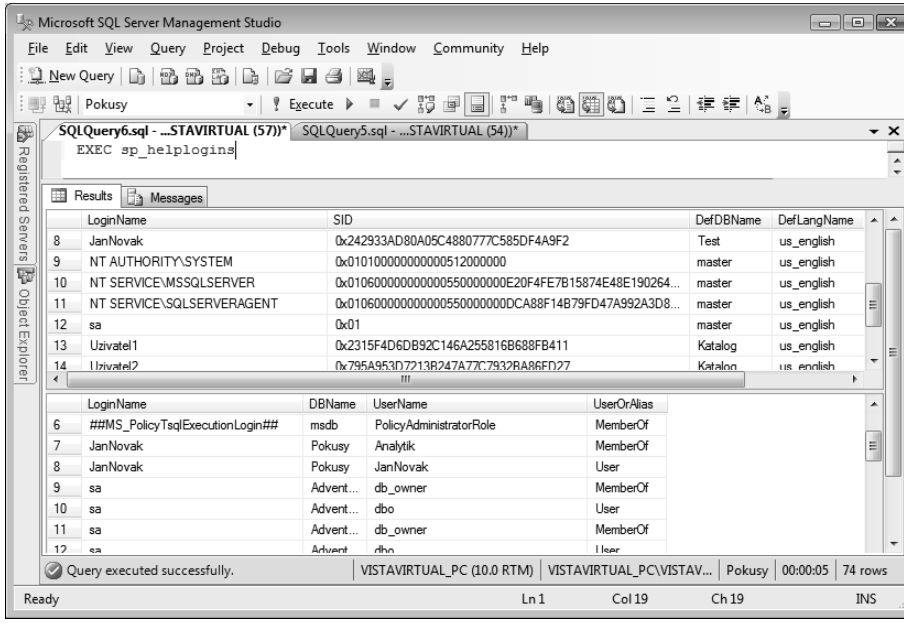
Například:

```
EXEC sp_helplogins
```

Schémata

SQL Server umožňuje existenci více schémat v jedné databázi, přičemž schémata existují nezávisle na uživatelích a každý uživatel má přednastavené schéma. Jméno schématu může nahradit jméno uživatele v objektu. Používání schémat usnadňuje správu databáze při migraci osob. Když se v SQL Server Management Studiu podíváte na seznam tabulek některé cvičné databáze, například AdventureWorks2008, zjistíte, že jsou ve tvaru:

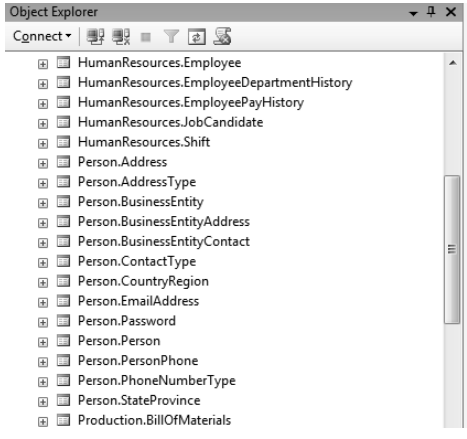
```
Schema.Nazev_tabulky
```



Obrázek 6.11 Výpis komplexních informací o uživatelích a jejich přiřazení do rolí

Například:

`Person.Address`



Obrázek 6.12 Fragment seznamu databázových tabulek cvičné databáze AdventureWorks2008

Zjednodušeně řečeno, schémata jsou definované skupiny vlastnických práv.

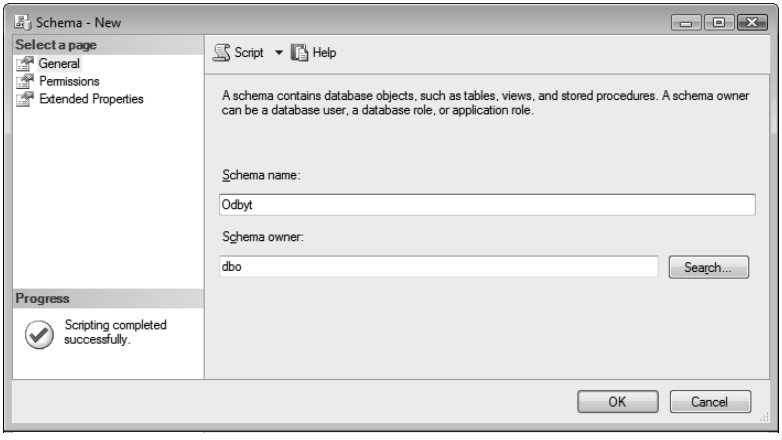
Upozornění: Je potřeba rozlišovat pojem „schéma“, kterým je definována skupina vlastnických práv, a pojem „schéma“ z hlediska databázové teorie. Přestože se používá stejný název, jde o dva významově úplně odlišné pojmy. Z hlediska databázové teorie představuje schéma opis struktury báze dat, definuje jednotlivé datové položky, databázové věty a logické vztahy mezi těmito větami. Schéma v tomto pojetí je vlastně opis logické a fyzické struktury dat.

Příklad pro vytvoření a použití schématu

Nové schéma lze vytvořit pomocí příkazu CREATE SCHEMA nebo v návrhovém dialogu nástroje SQL Server Management Studio. Ve vlastní databázi můžete vytvořit například schéma Odbyt, a to pomocí SQL příkazu:

```
CREATE SCHEMA Odbyt AUTHORIZATION dbo
```

Nebo vytvoříte nové schéma pomocí návrhového dialogu. Vlastníkem schématu bude zatím vlastník databáze.



Obrázek 6.12 Dialog pro vytvoření nového schématu v SQL Server Management Studiu

Ve schématu vytvořte databázovou tabulku, která bude patřit do nově vytvořeného schématu:

```
CREATE TABLE Odbyt.Pracovnici
(
    Id INT,
    Jmeno VARCHAR(50)
)
```

Při dotazování pomocí intuitivně zadaného příkazu:

```
SELECT * FROM Pracovnici;
```

dojde k chybě. V takovém případě je nutné zadat název tabulky včetně schématu:

```
SELECT * FROM Odbyt.Pracovnici;
```

Nastavení práv uživatele pro přístup k objektům databáze

Pro přidělování konkrétních práv pro přístup jednotlivých uživatelů k objektům databáze se používá příkaz GRANT. Privilegia je možné přidělovat jednak pro vytváření objektů, jednak pro přístup k objektům databáze. Zjednodušená syntaxe příkazu GRANT pro udělování privilegií pro vytváření objektů je:

```
GRANT { ALL | ukon [ ,...n ] }
  TO nazev_uctu [ ,...n ]
```

Konkrétně je možné přidělovat privilegia k vykonání následujících příkazů pro vytváření objektů:

CREATE DATABASE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE TABLE, CREATE VIEW, BACKUP DATABASE, BACKUP LOG

Syntaxe příkazu GRANT pro přístup k objektům v databázi je:

```
GRANT
    { ALL [ PRIVILEGES ] | opravneni [ ,...n ] }
    {
        [ ( sloupec [ ,...n ] ) ] ON { tabulka | pohled }
        | ON { tabulka | pohled } [ ( sloupec [ ,...n ] ) ]
        | ON { ulozena_procedura | rozsirena_procedura }
        | ON { uzivatelsky_definovana_funkce }
    }
TO nazev_uctu [ ,...n ]
[ WITH GRANT OPTION ]
[ AS { group | role } ]
```

Například našemu nově vytvořenému uživateli JanNovak můžeme přidělit privilegia pro vytváření tabulek a pohledů příkazem:

```
GRANT create table, create view  TO JanNovak;
```

Příkazem REVOKE je možné dříve přidělená práva uživateli odebrat:

```
REVOKE create table FROM JanNovak;
```

Transparentní šifrování

Z hlediska bezpečnosti je důležité také zabezpečení údajů proti odcizení. Ukrást databázi z předchozích verzí SQL Serveru z jinak nezabezpečeného serveru nebylo až tak složité. Stačilo odpojit databázi, nebo jen zastavit databázový server, překopírovat „mdf“ a „ldf“ soubory na svůj vlastní počítač, připojit databázi ke svojí instanci SQL serveru a útočník měl k dispozici veškeré údaje z databáze.

SQL Server 2008 umožňuje zvýšit bezpečnost databázových aplikací transparentním šifrováním databází, datových a logických souborů. Toto umožňuje získat organizacím požadované certifikáty, například pro správu osobních údajů, případně naplnit přísná kritéria a předpisy informační bezpečnosti.

Šifrování údajů je možné implementovat bez nutnosti jakýchkoliv změn v souvisejících aplikacích. Nová verze SQL Serveru umožňuje taktéž pokročilé auditování přístupu

do databáze, dotazování a modifikování údajů. Tento nástroj spolehlivě dokumentuje, které údaje, kdy a kým byly přečtené, případně modifikované. Podporuje symetrické i asymetrické šifrovací klíče a digitální certifikáty. Aktivováním vlastnosti Transparent Data Encryption šifrování u databáze bude zašifrovaný datový soubor i transakční log na disku.

Možnost ochrany údajů v databázových tabulkách byla k dispozici už ve verzi SQL Server 2005, ale vyžadovala si změny v aplikaci, přesněji spolupráci aplikační logiky. Sloupce zašifrované pomocí funkce Encrypt bylo nutné na úrovni aplikační logiky dešifrovat pomocí uložené procedury Decrypt. Navíc nad zašifrovanými sloupci nemohl optimalizátor vykonávání dotazů aplikovat žádné indexy ani statistiky. Takže ve většině případů se za zvýšenou bezpečnost platilo snížením výkonu.

Šifrování údajů lze implementovat přímo na úrovni databáze bez nutnosti jakýchkoliv změn v souvisejících aplikacích. Jsou šifrovány nejen údaje na discích, tady datové soubory s příponou MDF a redo log soubory s příponou LDF, ale při zálohování pomocí funkce Backup jsou zálohovány i údaje na záložních médiích.

Ukrást databázi z předchozích verzí SQL Serveru bylo poměrně jednoduché. Stačilo odpojit databázi, nebo jen zastavit databázový server, překopírovat „mdf“ a „ldf“ soubory na svůj vlastní počítač, nebo disk a připojit databázi ke svojí instanci SQL serveru. Při použití transparentního šifrování je nezbytné mít pro rozšifrování databáze klíč.

Vytvoření a správa klíčů

Se šifrováním úzce souvisí i správa klíčů. SQL Server 2008 podporuje i hardwarové bezpečnostní moduly pro správu klíčů a autentifikaci od různých firem. Konsolidace správy klíčů podstatně zjednodušuje správu šifrování údajů ve velkých víceserverových datových centrech. Šifrovací klíč může být sám chráněný heslem, servisním, nebo externím klíčem.

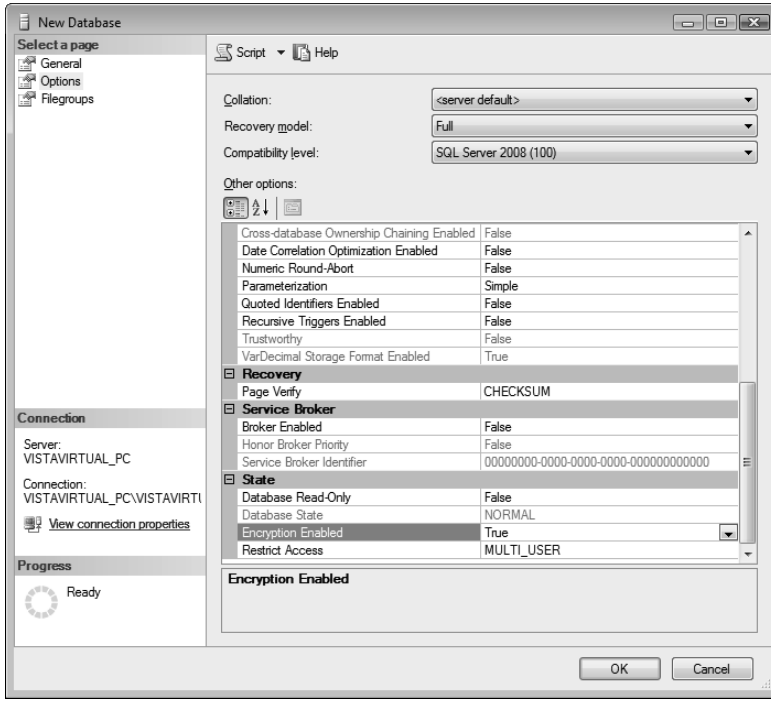
Externí šifrovací klíče

Ve verzi SQL Server 2005 musely být šifrovací klíče uložené přímo v databázích. Mnoho firem však využívá heterogenní bezpečnostní prostředí společné pro celou firmu. Ve verzi 2008 je podporované také poskytování šifrovacích klíčů od třetích stran, například RSA, NCipher, Safenet a podobně.

SQL Server 2008 podporuje symetrické a asymetrické šifrovací klíče a digitální certifikáty. Aktivováním vlastnosti **Transparent Data Encryption** šifrování u databáze bude zašifrovaný datový soubor i transakční log na disku. Nezašifrované jsou jen údaje v cache paměti. **Transparent Data Encryption** v principu šifruje, respektive dešifruje v reálném čase diskové I/O operace.

Pro aplikace přistupující k datům se nic nemění, přístup k údajům je řízený „grantováním“ práv. Pro šifrování databáze slouží speciální klíč „database encryption key“ (DEK), který je uložený v „boot“ záznamu databáze pro dostupnost dat i v době „recovery“. DEK je chráněný certifikátem uloženým v „master“ databázi. Data jsou šifrovaná použitím algoritmů AES128, AES192, AES256 nebo TripleDES (výchozím algoritmem je AES128).

Šifrování lze povolit již při vytváření nové databáze nastavením parametru **Encryption Enabled** na hodnotu **True**.



Obrázek 6.13 Povolení šifrování v dialogu pro nastavování vlastností databáze

Nebo pomocí příkazu:

```
ALTER DATABASE nazev_databaze SET ENCRYPTION ON
```

Například:

```
ALTER DATABASE Pokusy SET ENCRYPTION ON
```

Pro vytvoření MASTER KEY slouží příkaz:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'heslo'
```

Například:

```
USE master
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'nbusr123'
```

Nový certifikát se vytvoří například příkazem:

```
CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'Muj Certifikat'
```

Pro certifikát můžete vytvořit uživatele:

```
CREATE USER certUser FOR CERTIFICATE NasCertifikat
```

A přidělit mu práva například pro výběr údajů z tabulky:

```
GRANT SELECT ON zakaznici TO certUser
```

Vytvořte šifrovací klíč příkazem:

```
CREATE DATABASE ENCRYPTION KEY
       WITH ALGORITHM = { AES_128 | AES_192 | AES_256 | TRIPLE_DES_3KEY }
   ENCRYPTION BY SERVER
    {
                CERTIFICATE Encryptor_Name |
        ASYMMETRIC KEY Encryptor_Name
    }
[ ; ]
```

Například:

```
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE MyServerCert
```

Zapněte šifrování pro databázi, na kterou chcete šifrování uplatnit:

```
ALTER DATABASE Pokusy SET ENCRYPTION ON
```

Pro lepší přehlednost uvedeme znovu celý postup jako posloupnost příkazů:

```
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'nbusr123';
CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'Muj Certifikat'
go
USE Pokusy
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE MyServerCert
ALTER DATABASE Pokusy SET ENCRYPTION ON
GO
```

Stav šifrování lze ověřit pomocí systémového pohledu sys.dm_database_encryption_keys.

Auditování

Při zabezpečení databázového serveru jako integrální součásti informačního systému je důležitá i možnost auditování. SQL Server 2008 umožňuje pokročilé auditování přístupu do databáze, dotazování a modifikování údajů. Tento nástroj spolehlivě dokumentuje, které údaje, kdy a kým byly čtené nebo modifikované. Ve starší verzi SQL Server 2005 byly k dispozici všechny stavební bloky, ze kterých bylo možné vybudovat funkcionalitu auditování, ve verzi 2008 je auditování k dispozici jako hotové řešení.

K auditování se využívá funkcionalita SQL Server Extended Events. Hlavním pilířem řešení je objekt AUDIT. Výstup auditu je možné přesměrovat do souboru nebo protokolu

serverových událostí. Nad výsledky auditu můžeme pracovat s analytickými službami, případně zobrazovat výsledky pomocí reportovacích služeb.

Auditování si názorně předvedeme na jednoduchém příkladu. Předmětem příkladu bude audit, a to na dvou úrovních. Na úrovni serveru budou auditovány DBCC operace a na úrovni databázové tabulky budou auditovány příkazy SELECT.

Nejprve musíme vytvořit záznamy, v tomto případě vytvoříme v testovací databázi tabulku a naplníme ji údaji:

```
CREATE TABLE dbo.Mzdy
(
   jmeno varchar(25),
   mzda money
)

INSERT dbo.Mzdy VALUES ('Novák', 23000), ('Vopičková',  19200)
```

Postup je možné rozdělit do několika logicky navazujících úkolů:

1. Vytvoření objektu typu AUDIT pro server

Objekt typu audit je v podstatě úložiště pro ukládání výsledků auditu. Může to být buď soubor, nebo event log. V příkladě použijeme soubor, proto ještě před spuštěním příkazu pro vytvoření serverového auditu vytvořte na disku vhodný adresář. Objekt Audit bude v tomto případě vytvořený na úrovni serveru, proto musíme adresovat příkaz CREATE SERVER AUDIT do databázové tabulky MASTER.

```
USE master
GO
CREATE SERVER AUDIT MujAudit
TO FILE (FILEPATH = N'C:\Audit\')
WITH (ON_FAILURE = CONTINUE)
GO
```

Parametru pro objekt Audit si můžete zobrazit a nastavit i vizuálně pomocí nástroje SQL Server Management Studio. Příslušný dialog se otvírá v místní nabídce **Security → Audits**.

2. Povolení auditu

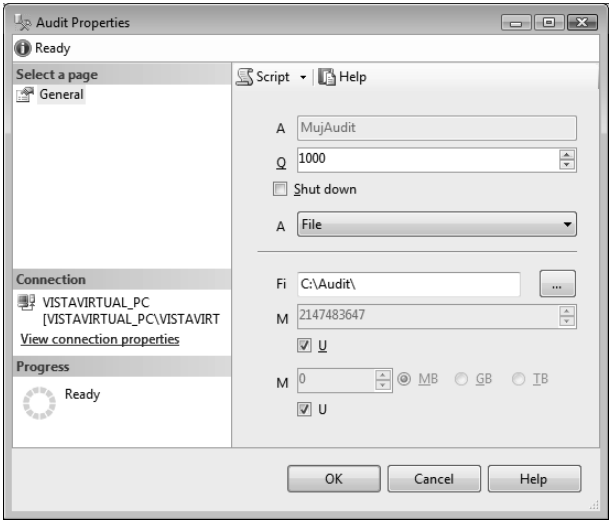
Po vytvoření objektu AUDIT je potřeba audit aktivovat. To můžete provést buď pomocí příkazu:

```
ALTER SERVER AUDIT MujAudit WITH (STATE=ON)
GO
```

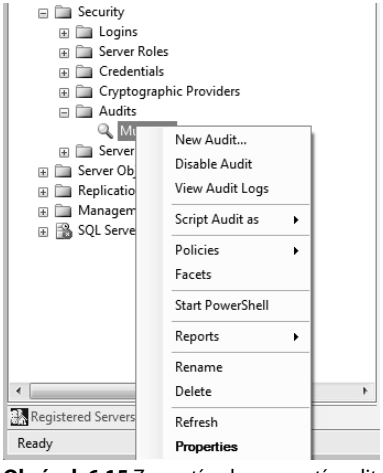
nebo prostřednictvím nástroje SQL Server Management Studio v místní nabídce **Security → Audits**.

3. Vytvoření specifikace auditu

Předmětem auditu mohou být akce na různé úrovni granularity. Audit může probíhat na úrovni databázového serveru, kdy jde například o vytváření uživatelských účtů, změny hesla a podobně, nebo na úrovni databáze, kdy jde o auditování dotazů do některých



Obrázek 6.14 Dialog pro nastavení parametrů objektu Audit

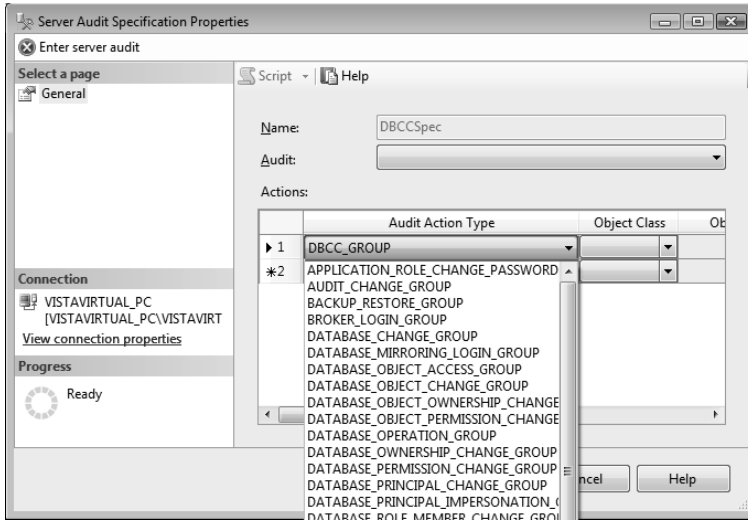


Obrázek 6.15 Zapnutí nebo vypnutí auditu

tabulek. Pomocí filtrování lze zaměřit audit tam, kam potřebujete. Můžete specifikovat, pro které uživatele nebo objekty chcete audit aktivovat. V následujícím příkladu vytvoříme specifikaci pro auditování DBCC příkazů:

```
CREATE SERVER AUDIT SPECIFICATION DBCCSpec
FOR SERVER AUDIT MujAudit
    ADD (DBCC_GROUP);
GO
ALTER SERVER AUDIT SPECIFICATION DBCCSpec WITH (STATE=ON)
GO
```

Specifikace auditu lze vytvářet také ve vizuálním návrhovém prostředí SQL Server Management Studio.



Obrázek 6.16 Vytvoření specifikace auditu

Podobně jako audit, i příslušné specifikace je třeba aktivovat.

Pro určitý audit je možné vytvořit serverové i databázové specifikace. Jako příklad specifikace databázového auditu si ukážeme auditování příkazu SELECT do tabulky Mzdy. Všimněte si, že specifikace databázového auditu se vytváří v příslušné databázi.

```
USE Test
GO
CREATE DATABASE AUDIT SPECIFICATION SelectMzdySpec
FOR SERVER AUDIT MujAudit
    ADD (SELECT ON dbo.Mzdy BY dbo)
GO
ALTER DATABASE AUDIT SPECIFICATION SelectMzdySpec WITH (STATE=ON)
GO
```

4. Testování auditu

Při vytváření prvního příkladu na určité téma, v tomto případě příklad na auditování, je důležité otestovat, jak příslušná funkcionalita pracuje. V případě auditu není nic jednoduššího, než vykonat akce, které jsou předmětem auditu, a přesvědčit se, jaké to zanechalo stopy. Audit na úrovni serveru je možné otestovat například pomocí příkazu pro kontrolu integrity systémové indexové tabulky.

```
DBCC UPDATEUSAGE (Test)
```

Podobně otestujeme také fungování databázového auditu pro příkaz SELECT do tabulky dbo.Mzdy.

```
SELECT * FROM dbo.Mzdy
```

O fungování auditování se přesvědčíte nahlédnutím do příslušného protokolu. Jeho umístění jste určili sami při vytváření objektu AUDIT.

Poznámka: Zápis do protokolu auditu je asynchronní operace, takže před nahlédnutím do protokolu auditu je potřeba několik sekund počkat.

Nejjednodušší způsob, jak prohlížet protokol auditu, je pomocí nástroje SQL Server Management Studio. K dispozici je však více možností, například SQL příkaz:

```
SELECT * FROM sys.fn_get_audit_file('C:\Audit\*',default,default);
```

Výsledkem takového univerzálního dotazu s hvězdičkou velké množství atributů, takže je trochu nepřehledný. Můžete si nechat vypsat jen ty atributy, které jsou pro vás důležité, například:

```
SELECT event_time, database_name, object_name, statement FROM sys.fn_get_audit_
file('C:\Audit\*',default,default);

event_time               database_name    object_name   statement
-------------------------------------------------------------------------------
2008-11-25 11:37:16.92   Test                           DBCC UPDATEUSAGE (Test)
2008-11-25 11:43:15.82   Test             Mzdy          SELECT * FROM dbo.Mzdy
```

5. Ukončení auditu

Audit představuje samozřejmě i určitou serverovou režii. Takže v případě, že auditování už splnilo svůj účel, je vhodné ho vypnout, případně také odstranit objekty s ním související. Takový úklid je obzvlášť dobrý ve studijní etapě po cvičných pokusech.

```
USE Test
GO
ALTER DATABASE AUDIT SPECIFICATION SelectMzdySpec WITH (STATE=OFF)
DROP DATABASE AUDIT SPECIFICATION SelectMzdySpec
-- DROP TABLE dbo.Mzdy

USE master
GO
ALTER DATABASE AUDIT SPECIFICATION DBCCSpec WITH (STATE=OFF)
DROP DATABASE AUDIT SPECIFICATION DBCCSpec
ALTER SERVER AUDIT MujAudit WITH (STATE = OFF)
DROP SERVER AUDIT MujAudit
```