1

Sistemske funkcije

SQL Server sadrži niz „sistemskih funkcija" pored uobičajenih funkcija u proizvodu. Neke od njih se koriste često i odmah je prilično je jasno kako treba da se koriste. Međutim, druge se ređe koriste i po prirodi su tajanstvenije.

U ovom dodatku, pokušaću da na kratak i koncizan način razjasnim upotrebu većine ovih funkcija.

NAPOMENA *Samo za vašu informaciju, u ranijim izdanjima, mnoge sistemske funkcije su se često nazivale* globalnim promenljivim. *To je pogrešan naziv, pa je Microsoft u nekoliko poslednjih izdanja pokušao da ga ispravi – menjanjem dokumentacije tako da se pravilnije nazivaju* sistemskim funkcijama. *Imajte na umu staru terminologiju ako neko staromodan (kao što sam ja) slučajno upotrebi ime globalne.*

T-SQL funkcije dostupne u SQL Serveru 2012 dele se na sledeće kategorije:

- Funkcije statistike sistema
- Agregatne funkcije
- Konfiguracione funkcije
- Kriptografske funkcije
- Kursorske funkcije
- Funkcije za datum i vreme
- Matematičke funkcije
- Funkcije metapodataka
- Rangirajuće funkcije

- Funkcije skupa redova
- Bezbednosne funkcije
- Funkcije za string
- Sistemske funkcije
- Funkcije za tekst i slike

NAPOMENA *Osim toga, postoji i operator* `OVER`*, koji, mada uglavnom služi kao alatka za rangiranje, može da se primeni i na druge vrste T-SQL funkcija (pogotovo na agregatne). Mada ga opisujem jedino u okviru funkcija za rangiranje, videćete da se pominje i na nekoliko drugih mesta u ovom dodatku.*

FUNKCIJE STATISTIKE SISTEMA

Prvi odeljak ovog dodatka sadrži sistemske funkcije vezane za operativno stanje i statistiku sistema. Primetićete da se većina njih razlikuje od drugih funkcija po tome što im imena počinju sa „@@"; to su sistemske funkcije koje su se ranije zvale globalne promenljive.

@@CONNECTIONS

Vraća broj pokušaja uspostavljanja konekcije od poslednjeg pokretanja SQL Servera.

To je ukupan broj svih pokušaja uspostavljanja konekcije nakon poslednjeg pokretanja SQL Servera. Ovde je bitno zapamtiti da je reč o pokušajima, a ne ostvarenim konekcijama, i da je reč o konekcijama a ne o korisnicima.

Svaki pokušaj da se uspostavi konekcija uvećava ovaj brojač bez obzira na to da li je konektovanje uspelo. Jedina začkoljica je u tome da je pokušaj konektovanja morao da dopre do servera. Ako pokušaj nije uspeo zbog NetLib razlika ili nekog drugog pitanja na mreži, vaš SQL Server neće ni znati da treba uvećati brojač – jedino se broji ako je server video pokušaj konektovanja. Nije bitno da li je pokušaj uspešan ili neuspešan.

Takođe je važno da se shvati da je reč o konekcijama a ne o pokušajima prijavljivanja. Zavisno od vaše aplikacije, možda ćete da pravite nekoliko konekcija ka vašem serveru, ali ćete verovatno samo jednom tražiti informacije od korisnika. Zaista, to radi čak i Query Analyzer. Kada kliknete za nov prozor, on automatski pravi još jednu konekciju na osnovu istih informacija za prijavljivanje.

NAPOMENA *Ovu, kao i niz drugih sistemskih funkcija, često je bolje koristiti u sistemskoj snimljenoj proceduri,* `sp_monitor`*. Ta procedura, jednom samom komandom, proizvodi informacije počevši od broja konekcija, CPU BUSY, sve do ukupnog broja pisanja koje je obavio SQL Server. Dakle, ako tražite osnovne informacije,* `sp_monitor` *je možda bolja – ali ako vam je potreban diskretan podatak kojim možete da manipulišete,* `@@CONNECTIONS` *daje lep, uredan, skalarni podatak.*

@@CPU_BUSY

Vraća vreme u milisekundama koje je procesor proveo aktivno radeći nakon poslednjeg pokretanja SQL Servera. Ovaj broj je zasnovan na rezoluciji sistemskog merača vremena – koja može da varira – pa, prema tome, varira u preciznosti.

Ovo je još jedna od funkcija vrste „nakon pokretanja servera". To znači da uvek možete da računate na to da se taj broj povećava kako se vaša aplikacija izvršava. Moguće je, na osnovu ovog broja, odrediti koji procenat procesora uzima vaš SQL Server. Realno posmatrajući, kad bi mi ta informacija bila užasno potrebna, radije bih posegao u Performance Monitor. U suštini, ovo je jedna od tih zaista elegantnih stvari za koje kažete „au, kako zanimljivo", ali u većini aplikacija nema neku praktičnu primenu.

sys.fn_virtualfilestats

Kada pozovete `sys.fn_virtualfilestats` saznaćete ukupna čitanja, pisanja i vreme čekanja na datoteku baze podataka od pokretanja tog primerka baze podataka. Za razliku od sistemskih funkcija koje ovde izgledaju kao da su u stvari globalne promenljive, ova funkcija se poziva kao obična funkcija sa tabelarnom vrednošću. Ona prihvata dva parametra – ID baze podataka i ID datoteke – a oba parametra mogu da budu `NULL` kao znak da želite nefiltrirane rezultate.

Ako ste prihvatili sve podrazumevane vrednosti za DB, `fileid 1` je datoteka podataka, a `fileid 2` je primarni dnevnik. Dakle, ako pozovete `sys.fn_virtualfilestats(NULL, 2)` dobićete statistiku za sve datoteke primarnog dnevnika. Dobijaju se podaci:

- **`Dbid`**: ID baze podataka za ovaj izlazni red
- **`FileID`**: ID datoteke
- **`TimeStamp`**: Broj sekundi rada vašeg primerka
- **`NumberReads`**: Ukupan broj čitanja ove datoteke od pokretanja primerka
- **`NumberWrites`**: Ukupan broj pisanja ove datoteke od pokretanja primerka
- **`BytesRead`**: Ukupan broj bajtova pročitanih iz ove datoteke
- **`BytesWritten`**: Ukupan broj bajtova upisanih u ovu datoteku
- **`IoStallMs`**: Broj milisekundi koje su procesi proveli u čekanju da se vrati UI

Broj čitanja i pisanja pružaju odličan utisak o obimu ulaza i izlaza, ali `IoStallMs` može da bude neprocenjivo za pronalaženje uskih grla performansi.

@@IDLE

Vraća vreme u milisekundama (na osnovu rezolucije sistemskog merača vremena) koje je SQL Server proveo besposlen nakon poslednjeg pokretanja.

Ovo možete da shvatite kao suprotno od `@@CPU_BUSY`. U suštini, saznajete koliko vremena je vaš SQL Server proveo ne radeći ništa. Ako ikome padne na pamet bilo kakva programske upotreba ovog podatka, neka mi pošalje e-poruku – voleo bih da čujem (ja ne mogu da se setim).

@@IO_BUSY

Vraća vreme u milisekundama (na osnovu rezolucije sistemskog merača vremena) koje je SQL Server proveo vršeći ulazne i izlazne operacije nakon poslednjeg pokretanja. Ova vrednost se vraća na nulu prilikom svakog pokretanja SQL Servera.

Ovo ne predstavlja neku posebnu nauku, i takođe po meni spada u kategoriju „bez prave programske primene".

@@PACK_RECEIVED i @@PACK_SENT

Vraćaju broj paketa koje je SQL Server primio sa mreže, odnosno poslao na mrežu nakon poslednjeg pokretanja.

Ovo su pre svega, alatke za istraživanje problema na mreži.

@@PACKET_ERRORS

Vraća broj grešaka mrežnih paketa do kojih je došlo na konekcijama vašeg SQL Servera nakon poslednjeg pokretanja SQL Servera.

Pre svega, alatka za istraživanje problema na mreži.

@@TIMETICKS

Vraća broj mikrosekundi po otkucaju. To varira među mašinama, još jedna od funkcija koje spadaju u kategoriju „bez prave programske primene."

@@TOTAL_ERRORS

Vraća broj grešaka čitanja/pisanja diska na koje je naišao SQL Server nakon poslednjeg pokretanja.

Nemojte ovo brkati sa greškama izvršavanja niti pomišljati da ima ikakve veze sa `@@ERROR`. Ovde je reč o problemima fizičkog U/I. To je još jedna od onih „bez prave programske primene". Prvenstvena primena bi bila u skriptovima za sistemsku dijagnostiku. Uglavnom bih za ovo pre upotrebio Performance Monitor.

@@TOTAL_READ and @@TOTAL_WRITE

Vraćaju ukupan broj čitanja diska, odnosno pisanja na disk koje je izvršio SQL Server nakon poslednjeg pokretanja.

Imena su ovde pomalo varljiva, jer ne uključuju čitanje iz keša – to je samo fizički U/I.

AGREGATNE FUNKCIJE

Agregatne funkcije se primenjuju na skupove zapisa a ne na pojedinačan zapis. Informacije iz više zapisa se obrađuju na određen način, a zatim se prikazuju kao odgovor u jednom zapisu. Agregatne funkcije se često koriste zajedno sa klauzulom `GROUP BY`.

Agregatne funkcije su:

- AVG
- CHECKSUM
- CHECKSUM_AGG
- COUNT
- COUNT_BIG
- GROUPING
- MAX
- MIN
- STDEV
- STDEVP
- SUM
- VAR
- VARP

U većini agregatnih funkcija mogu da se koriste ključne reči ALL ili DISTINCT. Argument ALL se podrazumeva i primeniće funkciju na sve vrednosti u expression, čak kad se ista vrednost javlja više puta. Argument DISTINCT znači da se vrednost uključuje u funkciju samo jednom, čak kad se javlja više puta.

Agregatne funkcije ne mogu da se ugnežđuju. Izraz expression ne može da bude podupit.

AVG

AVG vraća prosek vrednosti u expression. Sintaksa je sledeća:

```
AVG([ALL | DISTINCT] <expression>)
```

Izraz expression mora da sadrži numeričke vrednosti. NULL vrednosti se zanemaruju. Ova funkcija podržava operator OVER opisan u odeljku o rangirajućim funkcijama u ovom dodatku.

CHECKSUM

Ovo je osnovni heš algoritam koji se obično koristi za otkrivanje promena ili konzistentnosti u podacima. Ova konkretna funkcija može kao argument da primi izraz ili * (što znači da hoćete da se uključe sve kolone u svim spojenim tabelama). Osnovna sintaksa je:

```
CHECKSUM(<expression>, [...n] | * )
```

Obratite pažnju na to da redosled vaših izraza – ili u slučaju da koristite *, redosled spojeva – utiče na vrednost funkcije checksum, pa, na primer:

```
CHECKSUM(SalesOrderID, OrderDate)
```

ne bi dalo isti rezultat kao:

```
CHECKSUM(OrderDate, SalesOrderID )
```

Ova funkcija *NIJE* kompatibilna sa operatorom `OVER`.

NAPOMENA `CHECKSUM` *se razlikuje od ostalih agregatnih funkcija. One sve sjedinjuju više redova, a* `CHECKSUM` *radi samo nad jednim redom. Međutim, ona sjedinjuje (ili može da sjedinjuje) sve podatke u redu, pa se može braniti to što je ovde uvršćena. Međutim, moj pravi razlog što sam je ovde uključio, jeste da je mnogo lakše razumeti* `CHECKSUM_AGG` *ako ste već upoznati sa* `CHECKSUM`.

CHECKSUM_AGG

Kao `CHECKSUM`, ovo je osnovni heš koji se obično koristi za otkrivanje promena ili konzistentnosti podataka. Osnovna razlika je u tome što je `CHECKSUM` orijentisana na redove, dok je `CHECKSUM_AGG` orijentisana na kolone. Osnovna sintaksa je:

```
CHECKSUM_AGG( [ALL | DISTINCT] <expression>)
```

Vrednost izraza može da bude praktično bilo šta, uključujući, ako hoćete, sastavljanje kolona (samo ne zaboravite konverziju, ako je potrebna); međutim, ne zaboravite da je bitan redosled izraza, pa ako spajate, `Col1 + Col2` nije isto što i `Col2 + Col1`.

COUNT

`COUNT` vraća broj elemenata u `expression`. Tip vraćene vrednosti je `int`. Sintaksa je sledeća:

```
COUNT
(
   [ALL | DISTINCT] <expression> | *
)
```

Izraz ne može biti tipa `uniqueidentifier`, `text`, `image`, ili `ntext`. Argument `*` vraća broj redova u tabeli; ne eliminiše duplikate niti `NULL` vrednosti.

Ova funkcija podržava operator `OVER` opisan u odeljku o rangirajućim funkcijama ovog dodatka.

COUNT_BIG

`COUNT_BIG` vraća broj elemenata u grupi. Veoma je slična upravo opisanoj funkciji `COUNT`, osim što je vraćena vrednost tipa `bigint`. Sintaksa je sledeća:

```
COUNT_BIG
(
   [ALL | DISTINCT ] <expression> | *
)
```

Isto kao `COUNT`, ova funkcija podržava operator `OVER` opisan u odeljku o rangirajućim funkcijama ovog dodatka.

GROUPING

GROUPING dodaje još jednu kolonu na izlaz naredbe SELECT. Funkcija GROUPING se koristi zajedno sa CUBE ili ROLLUP da bi se razlikovale normalne NULL vrednosti od onih dobijenih kao rezultat operacija CUBE i ROLLUP. Sintaksa je:

```
GROUPING (<column name>)
```

GROUPING se koristi jedino u SELECT listi. Njen argument je kolona koja se koristi u klauzuli GROUP BY i u kojoj treba proveriti postojanje NULL vrednosti.

Ova funkcija podržava operator OVER opisan u odeljku o rangirajućim funkcijama ovog dodatka.

MAX

Funkcija MAX vraća najveću vrednost iz expression. Sintaksa je sledeća:

```
MAX([ALL | DISTINCT] <expression>)
```

MAX zanemaruje sve NULL vrednosti.

Ova funkcija podržava operator OVER opisan u odeljku o rangirajućim funkcijama ovog dodatka.

MIN

Funkcija MIN vraća najmanju vrednost iz expression. Sintaksa je sledeća:

```
MIN([ALL | DISTINCT] <expression>)
```

MIN zanemaruje NULL vrednosti.

Ova funkcija podržava operator OVER opisan u odeljku o rangirajućim funkcijama ovog dodatka.

STDEV

Funkcija STDEV vraća standardnu devijaciju svih vrednosti u expression. Sintaksa je sledeća:

```
STDEV(<expression>)
```

STDEV zanemaruje NULL vrednosti.

Ova funkcija podržava operator OVER opisan u odeljku o rangirajućim funkcijama ovog dodatka.

STDEVP

Funkcija STDEVP vraća standardnu devijaciju za populaciju svih vrednosti u expression. Sintaksa je sledeća:

```
STDEVP(<expression>)
```

STDEVP zanemaruje NULL vrednosti i podržava operator OVER opisan u odeljku o rangirajućim funkcijama ovog dodatka.

SUM

Funkcija SUM vraća zbir svih vrednosti u expression. Sintaksa je sledeća:

```
SUM([ALL | DISTINCT] <expression>)
```

SUM zanemaruje NULL vrednosti. Ova funkcija podržava operator OVER opisan u odeljku o rangirajućim funkcijama ovog dodatka.

VAR

Funkcija VAR vraća varijansu svih vrednosti u expression. Sintaksa je sledeća:

```
VAR(<expression>)
```

VAR zanemaruje NULL vrednosti. Ova funkcija podržava operator OVER opisan u odeljku o rangirajućim funkcijama ovog dodatka.

VARP

Funkcija VARP vraća varijansu za populaciju svih vrednosti u expression. Sintaksa je sledeća:

```
VARP(<expression>)
```

VARP zanemaruje NULL vrednosti. Ova funkcija podržava operator OVER opisan u odeljku o rangirajućim funkcijama ovog dodatka.

ANALITIČKE FUNKCIJE

SQL Server 2012 sadrži jednu novu klasu agregatnih funkcija koje služe kao pomoć za vršenje statističke analize. To su agregati, ali tamo gde običan agregat (kao što je SUM) može da vrati samo jednu vrednost za redove nad kojima je primenjen, ove mogu da vrate više redova. One su posebno korisne za izračunavanje pokretnih proseka ili kumulativnih zbireva.

CUME_DIST

```
CUME_DIST( )
    OVER ( [ partition_by_clause ] order_by_clause )
```

Vraća procenat (broj između 0 i 1) redova unutar oblasti (partition) čija je vrednost (vrednost se bira u klauzuli ORDER BY) manja ili jednaka tekućoj vrednosti.

FIRST_VALUE, LAST_VALUE

```
[ FIRST_VALUE | LAST_VALUE ] ( [scalar_expression )
OVER (
    [ partition_by_clause ] order_by_clause [ rows_range_clause ]
)
```

Prva i poslednja vrednost su nešto za čije izračunavanje su SQL programeri odavno smišljali trikove, a sada u SQL-u 2012 konačno imate direktnu funkciju koja vam ih pronalazi. To je odlično jer je ranije jedini način da se do njih dođe zahtevao (u nekoj fazi) spoj sa samim sobom. Ako ste propustili poglavlje 4, spoj sa samim sobom je kada jednu tabelu spajate sa njom samom; u svakom slučaju, spojevi su skupi, a ova funkcija nije toliko.

Obratite pažnju na to da je obavezna samo klauzula `ORDER BY`; izostavljanjem ostale dve klauzule dobija se najprostiji slučaj, kad hoćete jedino prvu vrednost celog skupa rezultata. Kada dodate klauzulu `PARTITION BY` možete da dobijete prvu vrednost svake oblasti (partition), a klauzula `ROWS RANGE` može još da suzi vaše rezultate tako što vam omogućava da suzite raspon redova unutar oblasti iz kojih birate prvu ili poslednju vrednost.

LAG, LEAD

```
[LAG | LEAD] (scalar_expression [,offset] [,default])
    OVER ( [ partition_by_clause ] order_by_clause )
```

Dok vam `FIRST_VALUE` i `LAST_VALUE` dopuštaju da nađete red na nekom kraju vašeg skupa rezultata, `LAG` i `LEAD` mogu da vam pronađu red koji se nalazi na nekoliko koraka ispred ili iza tekućeg reda. To može da bude korisno, recimo, za izračunavanje delte po vremenu; ako uzmete `LAG`, u svakom redu možete znati vrednost prethodnog reda, pa možete da izračunate priraštaj vrednosti.

Dok je klauzula `OVER` identična kao na drugim mestima, ostali parametri nisu toliko jasni sami po sebi.

- `scalar_expression`: vrednost iz odmaknutog reda koju hoćete da vratite.
- `offset`: koliko redova pre (za `LAG`) ili posle (za `LEAD`) hoćete da gledate; podrazumeva se 1.
- `default`: vrednost koju treba vratiti ako ne postoji red na navedenom pomeraju. To mora da bude implicitno konvertibilno u tip podatka `scalar_expression`.

PERCENTILE_CONT, PERCENTILE_DISC

```
 [ PERCENTILE_CONT | PERCENTILE_DISC ] ( numeric_literal )
WITHIN GROUP ( ORDER BY order_by_expression [ ASC | DESC ] )
OVER ( [ <partition_by_clause> ] )
```

Ove funkcije vraćaju vrednost reda u percentilu navedenom kao `numeric_literal` unutar opisane grupe. `PERCENTILE_DISC` pronalazi stvarnu vrednost reda – koji postoji najbliže granici percentila – i vraća ga, dok `PERCENTILE_CONT` pretpostavlja ravnomernu distribuciju i vraća vrednost na koju bi percentil pao (čak i da ta vrednost ne postoji u grupi).

Ove funkcije možete da zamislite kao da koriste funkciju `CUME_DIST`. `PERCENTILE_DISC` pronalazi i vraća red sa najnižim `CUME_DIST` ispod `numeric_literal`.

PERCENT_RANK

```
PERCENT_RANK( )
OVER ( [ partition_by_clause ] order_by_clause )
```

Izračunava procenat u koji pada rang svake vrednosti. To je nešto kao `CUME_DIST`, koji vam daje procenat redova koji padaju na tu vrednost ili ispod nje, ali uz jedan dodatak; identični redovi umanjuju broj raspoloživih različitih rangova, tako da vam `PERCENT_RANK` kaže koji procenat redova pada ispod ovog ranga. Pa ako u jednoj koloni postoje samo tri različite vrednosti, a najniža vrednost predstavlja polovinu svih vrednosti, funkcija `CUME_DIST` bi za nju vratila 0.5 („polovina svih vrednosti je jednaka ili manja od ove vrednosti"), dok bi funkcija `PERCENT_RANK` vratila nula („nijedan od redova nije ispod ove vrednosti"). Za srednju vrednost bi `PERCENT_RANK` uvek vratila 0.5, a za najvišu 1, bez obzira na to koliko je redova u svakom rangu.

KONFIGURACIONE FUNKCIJE

Dakle, siguran sam da ćete se potpuno iznenaditi (dobro, ne baš…), ali konfiguracione funkcije su one koje vas obaveštavaju o opcijama koje su odabrane na trenutnom serveru, odnosno bazi podataka.

@@DATEFIRST

Vraća numeričku vrednost koja odgovara danu u nedelji koji sistem smatra prvim danom nedelje.

U SAD se podrazumeva `7`, a to je nedelja (Sunday). Vrednosti se konvertuju na sledeći način:

- `1`: Monday (ponedeljak – prvi dan u većini sveta)
- `2`: Tuesday (utorak)
- `3`: Wednesday (sreda)
- `4`: Thursday (četvrtak)
- `5`: Friday (petak)
- `6`: Saturday (subota)
- `7`: Sunday (nedelja)

Ovo može da bude zaista praktično za pitanja lokalizacije, tako da možete pravilno da postavite svaki kalendar i ostale informacije koje zavise od dana u nedelji.

NAPOMENA *Za menjanje ovog parametra, upotrebite funkciju* `SET DATEFIRST`.

@@DBTS

Vraća poslednji upotrebljeni vremenski pečat za trenutnu bazu podataka.

Na pvi pogled, ovo strašno liči na `@@IDENTITY` po tome što vam daje šansu da vidite poslednju vrednost koju je sistem odredio (ovde je reč o poslednjem vremenskom pečatu umesto o poslednjoj vrednosti identiteta). Potrebno je obratiti pažnju na sledeće:

- Vrednost se menja za svaku promenu u bazi podataka, ne samo za tabelu na kojoj vi radite.
- Reflektuju se *sve* promene vremenskog pečata u bazi podataka, ne samo one za trenutnu konekciju.

Pošto ne možete da računate na to da je ovo zaista poslednja vrednost koju ste vi upotrebili (neko drugi je mogao da uradi nešto što je menja), ja lično ne vidim neku naročitu praktičnu korist od ove funkcije.

@@LANGID i @@LANGUAGE

Vraća ID, odnosno ime jezika koji se trenutno koristi.

To može da bude praktično da biste utvrdili da li je proizvod instaliran u lokalizovanom okruženju, pa ako jeste, koji jezik se podrazumeva.

Potpun spisak jezika koje SQL Server trenutno podržava dobićete od sistemske snimljene procedure, `sp_helplanguage`.

@@LOCK_TIMEOUT

Vraća trenutno preostalu količinu vremena u milisekundama dok sistem ne obustavi čekanje na blokiran resurs.

Ako je neki resurs (stranica,red, tabela, bilo šta) blokiran, vaš proces staje i čeka da se blokiranje prekine. Ovde se vidi koliko dugo će vaš proces još da čeka dok ne otkaže naredbu.

Podrazumevano vreme čekanja je `0` (što znači beskonačno) ukoliko ga niko nije menjao na nivou sistema (pomoću `sp_configure`). Bez obzira na postavljenu sistemsku podrazumevanu vrednost, od ove globalne promenjive dobijate `-1` ukoliko niste ručno postavili vrednost za trenutnu konekciju pomoću `SET LOCK_TIMEOUT`.

@@MAX_CONNECTIONS

Vraća maksimalno dozvoljen broj istovremenih korisničkih konekcija na vašem SQL Serveru.

Morate da znate da ovo ne znači isto što biste našli pod svojstvom Maximum Connections u Management Console. Ovo se odnosi na licenciranje i pokazaće veoma visok broj ako ste odabrali licenciranje „per seat".

NAPOMENA *Obratite pažnju na to da stvarno dozvoljeni broj korisničkih konekcija takođe zavisi od verzije SQL Servera koju koristite i od ograničenja vaših aplikacija i hardvera.*

@@MAX_PRECISION

Vraća nivo preciznosti trenutno postavljen za decimalne i numeričke tipove podataka.

Podrazumeva se 38 mesta, ali vrednost može da se menja opcijom `/p` kod pokretanja SQL Servera. Opcija `/p` može da se doda kada se SQL Server pokreće sa komandne linije, ili da se doda u parametre za Startup servisa MSSQLServer u apletu Windowsa 2000, 2003, XP, ili 2008.

@@NESTLEVEL

Vraća trenutni nivo ugnežđavanja za ugnežđene snimljene procedure.

Prva snimljena procedura koja se izvršava ima `@@NESTLEVEL` jednak `0`. Ako ta snimljena procedura pozove drugu, za tu drugu snimljenu proceduru se kaže da je ugnežđena u prvoj (pa se `@@NESTLEVEL` uvećava na vrednost `1`). Isto tako, druga snimljena procedura može da pozove treću, i tako dalje do maksimalna 32 nivoa u dubinu. Ako premašite granicu od 32 nivoa dubine, osim što će se transakcija prekinuti, trebalo bi i da revidirate dizajn svoje aplikacije.

@@OPTIONS

Vraća informacije o opcijama primenjenim pomoću komande `SET`.

Pošto vam se vraća samo jedna vrednost, a mogle su da budu postavljene mnoge opcije, SQL Server koristi binarne indikatore da označi koje vrednosti su postavljene. Da biste ispitali da li je postavljena opcija koja vas zanima, morate da upotrebite vrednost opcije zajedno sa operatorom za bitove. Na primer:

```
IF (@@OPTIONS & 2)
```

Ako je ovo `True` (tačno), znaćete da je za trenutnu konekciju uključeno `IMPLICIT_TRANSACTIONS`. Vrednosti su prikazane u tabeli B-1.

TABELA B-1: Vrednosti bitmaski za @@OPTIONS

BIT	SET OPCIJA	OPIS
1	`DISABLE_DEF_CNST_CHK`	Prethodno umesto odgođenog proveravanja ograničenja.
2	`IMPLICIT_TRANSACTIONS`	Transakcija se pokreće implicitno čim se izvrši naredba.
4	`CURSOR_CLOSEON_COMMIT`	Kontroliše ponašanje kursora nakon operacije `COMMIT`.
8	`ANSI_WARNINGS`	Upozorava na odsecanje i `NULL` u agregatima.
16	`ANSI_PADDING`	Kontroliše dopunjavanje promenljivih fiksne dužine.
32	`ANSI_NULLS`	Određuje rukovanje null vrednostima kada se koriste operatori jednakosti.

BIT	SET OPCIJA	OPIS
64	ARITHABORT	Prekida upit ako se prilikom njegovog izvršavanja javi prekoračenje ili deljenje nulom.
128	ARITHIGNORE	Vraća NULL kada se prilikom izvršavanja upita javi prekoračenje ili deljenje nulom.
256	QUOTED_IDENTIFIER	Razlikuje jednostruke i dvostruke navodnike prilikom procenjivanja izraza.
512	NOCOUNT	Isključuje poruku o broju pogođenih redova koja se vraća na kraju svake naredbe.
1024	ANSI_NULL_DFLT_ON	Menja ponašanje sesije na primenu ANSI kompatibilnosti za null vrednosti. Kolone koje se prave u novim tabelama ili se dodaju na postojeće bez eksplicitno podešene opcije za null, definišu se da dozvoljavaju null. Međusobno isključivo sa ANSI_NULL_DFLT_OFF.
2048	ANSI_NULL_DFLT_OFF	Menja ponašanje sesije da ne primenjuje ANSI kompatibilnosti za null vrednosti. Nove kolone definisane bez eksplicitno podešene opcije za null se definišu da ne dozvoljavaju null. Međusobno isključivo sa ANSI_NULL_DFLT_ON.
4096	CONCAT_NULL_YIELDS_NULL	Vraća NULL kada se NULL spaja sa stringom.
8192	NUMERIC_ROUNDABORT	Generiše grešku kada u izrazu dođe do gubitka preciznosti.

@@REMSERVER

Vraća vrednost servera (kakva se javlja u zapisu prijavljivanja) koji je pozvao snimljenu proceduru.

Koristi se jedino u snimljenim procedurama. Ova funkcija je praktična kada hoćete da se snimljena procedura ponaša različito u zavisnosti od toga sa kojeg udaljenog servera (često neke geografske lokacije) je stigao poziv.

@@SERVERNAME

Vraća ime lokalnog servera sa kojeg se skript izvršava.

Ako ste instalirali više primeraka SQL Servera (dobar primer bi bila hosting kompanija koja koristi zasebne instalacije SQL Servera za svakog klijenta), @@SERVERNAME vraća informaciju o imenu lokalnog servera, prikazanu u tabeli B-2, ako ime lokalnog servera nije menjano od postavke.

TABELA B-2: Informacije o imenu lokalnog servera koje vraća @@SERVERNAME

PRIMERAK	SERVER INFORMATION
Podrazumevani primerak	`<imeservera>`
Imenovani primerak	`<imeservera\imeprimerka>`
Virtuelni server – podrazumevani primerak	`<imevirtuelnogservera>`
Virtuelni server – imenovani primerak	`<imevirtuelnogservera\imeprimerka>`

@@SERVICENAME

Vraća ime ključa baze Registry pod kojim se izvršava SQL Server. To će biti MSSQLService ako je reč o podrazumevanom primerku SQL Servera, ili odgovarajuće ime primerka.

@@SPID

Vraća ID serverskog procesa (SPID) tekućeg korisničkog procesa.

To je isti ID procesa koji vidite ako izvršite `sp_who`. Ovde je zgodno što možete da saznate `SPID` vaše tekuće konekcije, pa administrator baze podataka može da ga upotrebi za nadgledanje i, po potrebi, prekidanje tog zadatka.

@@TEXTSIZE

Vraća trenutnu vrednost opcije `TEXTSIZE` naredbe `SET`, koja određuje maksimalnu dužinu, u bajtovima, koju naredba `SELECT` može da vrati kada radi sa tekstualnim ili slikovnim podacima.

Podrazumeva se 4.096 bajtova (4 KB). Ovu vrednost možete da menjate pomoću naredbe `SET TEXTSIZE`.

@@VERSION

Vraća tekuću verziju SQL Servera kao i tip procesora i arhitekturu OS-a.

Na primer:

```
SELECT @@VERSION
```

daje:

```
------------------------------------------------------------------------
Microsoft SQL Server 2012 RC0 - 11.0.1750.32 (X64)
    Nov  4 2011 17:54:22
    Copyright (c) Microsoft Corporation
    Enterprise Evaluation Edition (64-bit) on Windows NT 6.1 <X64>
(Build 7601: Service Pack 1) (Hypervisor)

(1 row(s) affected)
```

Nažalost, ove informacije se ne vraćaju ni u kakvom structurisanom rasporedu polja, pa morate da raščlanjujete ako hoćete da ispitate neku konkretnu informaciju.

Razmotrite radije korišćenje sistemske snimljene procedure `xp_msver` – ona vraća informacije na takav način da možete lakše da izdvajate pojedine informacije iz rezultata.

FUNKCIJE ZA KONVERZIJU

Zar ne bi bilo praktično kad bi SQL Server mogao slobodno da konvertuje podatke između bilo koja dva tipa? Čuo sam programere početnike kako vrište od frustracije jer je njihova želja trebalo da bude očigledna („Sastavio sam taj broj u string… pa NARAVNO da sam hteo da se konvertuje u string!"), iako SQL zahteva da budete malo eksplicitniji. U stvari, ako uzmete u obzir različite kulture retko kad je nešto očigledno. Na primer, datum „01/02/03" jednom Amerikancu znači jedno (2. januar, 2003), a većini ostalog sveta nešto sasvim drugo (1. februar, 2003). Funkcije u sledećem odeljku pomažu vam da te i druge konverzije budu eksplicitnije i jasnije ali vas ne preopterećuju detaljima ako ne morate da ih koristite.

CAST i CONVERT

Ove dve funkcije imaju sličnu funkcionalnost po tome što obe konvertuju jedan tip podatka u drugi. `CAST` je jednostavniji, podrazumeva sve što se može, dok `CONVERT` zahteva da definišete izvorni i odredišni format kada `CAST` ne bi bio deterministički.

- Sa `CAST`

```
CAST(<expression> AS <data type>)
```

- Sa `CONVERT`:

```
CONVERT (<data type>[(<length>)], <expression> [, <style>])
```

  Gde se `style` odnosi na stil formata podatka prilikom konvertovanja u znakovni tip podataka.

PARSE

`PARSE` veoma liči na `CAST`, ali jedini izvor je string vrednost i svestan je kulture. `CAST` uvek koristi trenutni sistemski parametar kulture, dok `PARSE` dopušta da navedete parametar kulture koji treba primeniti u toj konverziji (kao podrazumevanu vrednost primeniće sistemski parametar).

```
PARSE ( string_value AS data_type [ USING culture ] )
```

Dakle, za primer iz uvoda bi kôd:

```
PARSE ( '01/02/03' AS DATE USING 'en-US' )
```

vratio datum kao January 2nd, 2003, dok bi kôd

```
PARSE ('01/02/03' AS DATE USING 'fr-FR' )
```

vratio datumsku vrednost kao February 1st, 2003.

TRY_CONVERT

`TRY_CONVERT` radi tačno isto što i `CONVERT`, uz jedan izuzetak. Dok `CONVERT` izbacuje grešku ako je konvertovanje nemoguće (pokušajte, na primer da konvertujete string `'99/99/NONE'` u datum), `TRY_CONVERT` vraća konvertovanu vrednost ako konvertovanje uspe, a `NULL` ako ne uspe i ne izbacuje nikakvu grešku.

```
TRY_CONVERT ( data_type [ ( length ) ], expression [, style ] )
```

TRY_PARSE

Kao i `TRY_CONVERT`, `TRY_PARSE` pokušava da raščlani podneti string u ciljni tip podatka, a ostaje miran i vraća vrednost `NULL` ako ne uspe.

```
TRY_PARSE ( string_value AS data_type [ USING culture ] )
```

KRIPTOGRAFSKE FUNKCIJE

Ove funkcije pomažu da se podrži šifrovanje, dešifrovanje, digitalno potpisivanje i provera valjanosti digitalnih potpisa. Neke od njih su bile nove u SQL Serveru 2008, a neke su došle još sa SQL Serverom 2005; SQL Server 2012 nije mnogo dodao u ovom domenu. Primetićete da, iz aspekta opšte upotrebe, postoje duplikati većine funkcija, koji se razlikuju po tome što jedna podržava simetričan ključ, a duplikat (obično u imenu sadrži `Asym`) podržava asimetričan ključ.

Sada, možda se pitate „šta će mi to?" Odgovori su raznovrsni koliko i mogućnosti primene SQL Servera. Najkraći odgovor je ovaj: ako ikad šaljete ili primate podatke koje biste hteli da zaštitite tokom prenosa. Na primer, pošto SQL Server podržava krajnje HTTP tačke, pa samim tim i hosting vlastitih veb servisa, možda ćete poželeti od nekog klijenta vašeg veb servisa da primate šifrovane informacije ili mu ih vraćate. Možda jedan prostiji primer: da ste jednostavno rešili da šifrujete podatke u svojoj bazi podataka, a sada treba da ih opet izvučete na koristan način.

AsymKey_ID

Ako je dato ime asimetričnog ključa, ova funkcija vraća `int` koji predstavlja odgovarajući ID iz baze podataka. Sintaksa je jednostavna:

```
AsymKey_ID('<Asymmetric Key Name>')
```

Da biste koristili ovu funkciju morate imati dozvolu za ključ o kojem je reč.

Cert_ID

Slično `AsymKey_ID`, ova funkcija vraća ID koji odgovara imenu jednog imena certifikata. Sintaksa je jednostavna:

```
Cert_ID('<Certificate Name>')
```

Da biste koristili ovu funkciju morate imati dozvolu za certifikat o kojem je reč.

CertProperty

Omogućava da dođete do različitih svojstava datog certifikata (identifikovanog pomoću ID-a certifikata). Moguća svojstva su datum početka, datum prestanka važenja, ime izdavaoca certifikata, serijski broj, bezbednosni ID („SID" koji može da se vrati i kao string) i predmet certifikata (ko ili šta se certifikuje). Sintaksa izgleda ovako:

```
CertProperty ( <Cert_ID> ,
    '<Expiry Date>' | '<Start Date>' | '<Issuing Authority>' |
    '<Certificate Serial Number>' | '<Subject>' | '<Security ID>' |
    '<SID as a String>'  )
```

Vraćeni tipovi podataka se razlikuju zavisno od konkretnog svojstva koje tražite (`datetime`, `nvarchar`, ili `varbinary`, kako je prikladno).

DecryptByAsmKey

Kao što možete naslutiti iz imena, ova funkcija dešifruje komad podataka pomoću jednog asimetričnog ključa. Potreban je ključ (po ID-u), šifrovani podaci (bilo kao literalan string ili kao promenljiva koja može da se ograniči na string) i lozinka za šifrovanje asimetričnog ključa (ako je upotrebljena kada se ključ pravio). Sintaksa je sasvim jednostavna:

```
DecryptByAsymKey(<Asymmetric Key ID>, {'<encrypted string>'|<string variable>}
    [, '<password>'])
```

Imajte na umu da, ako je korišćena lozinka da bi se napravio asimetričan ključ, ista lozinka će biti neophodna da bi se pomoću tog ključa pravilno dešifrovali podaci.

DecryptByCert

Ovo je u suštini isto što i `DecryptByAsmKey`, osim što umesto asimetričnog ključa očekuje certifikat. Kao `DecryptByAsmKey`, ova funkcija dešifruje komad podataka pomoću jednog ključa. Potreban je certifikat (po ID-u), šifrovani podaci (bilo kao literalan string ili kao promenljiva koja može da se ograniči na string) i lozinka za dalje šifrovanje podataka (ako je bila upotrebljena). Sintaksa izgleda skoro ista kao za `DecryptByAsymKey`:

```
DecryptByCert(<Certificate ID>, {'<encrypted string>'|<string variable>}
    [, '<password>'])
```

I ovde, ako je korišćena lozinka da bi se podaci šifrovali ista će biti potrebna da bi se pravilno dešifrovali.

DecryptByKey

Kao njeni asimetrični i certifikatski srodnici, ova funkcija dešifruje komad podataka pomoću jednog ključa. Ovde je različito ne samo to da funkcija očekuje simetričan ključ (umesto drugih vrsta ključeva), već da takođe očekuje da taj ključ već bude „otvoren" (pomoću komande `OPEN SYMMETRIC KEY`). Osim toga, koristi se sasvim slično, sa šifrovanim podacima (bilo kao literalan string ili kao promenljiva koja može da se ograniči na string) ubačenim kao parametar i, u ovom slučaju, heš ključ neobavezno prihvaćen za proveru verodostojnosti:

```
DecryptByKey({'<encrypted string>'|<string variable>},
    [<add authenticator value>, '<authentication hash>'|<string variable>])
```

Pazite da, ako dodajete vrednost za proveru autentičnosti sabiranjem (u formatu `int`), ta vrednost mora da odgovara vrednosti upotrebljenoj kada je string šifrovan, a morate takođe da obezbedite heš vrednost koja odgovara hešu obezbeđenom u vreme šifrovanja.

DecryptByPassPhrase

Kao što se naslućuje iz imena, ova funkcija dešifruje podatke koji su šifrovani ne pomoću zvaničnog ključa, već pomoću višedelne lozinke (passphrase). Osim što prihvata višedelnu lozinku kao parametar umesto da očekuje otvoren ključ, `DecryptByPassPhrase` radi skoro isto kao `DecryptByKey`:

```
DecryptByPassPhrase({'<passphrase>'|<string variable>},
    {'<encrypted string>'|<string variable>},
    [<add authenticator value>, '<authentication hash>'|<string variable>])
```

Kao i za `DecryptByKey`, ako dodajete vrednost za proveru autentičnosti sabiranjem (u formatu `int`), ta vrednost mora da odgovara vrednosti upotrebljenoj kada je string šifrovan, a morate takođe da obezbedite heš vrednost koja odgovara hešu obezbeđenom u vreme šifrovanja.

EncryptByAsmKey

Šifruje komad podataka pomoću asimetričnog ključa. Potreban je ključ (po ID-u) i podaci koje treba šifrovati (bilo kao literalan string ili kao promenljiva koja može da se ograniči na string). Sintaksa je sasvim jednostavna:

```
EncryptByAsymKey(<Asymmetric Key ID>,
{'<string to encrypt>'|<string variable>})
```

Imajte na umu da, ako je upotrebljena lozinka kada se asimetričan ključ dodavao bazi podataka, ista lozinka će biti neophodna za pravilno dešifrovanje podataka šifrovanih pomoću tog ključa.

EncryptByCert

Ovo je u suštini isto što i `EncryptByAsmKey`, osim što umesto asimetričnog ključa očekuje certifikat. Kao `EncryptByAsmKey`, ova funkcija šifruje komad podataka pomoću datog ključa. Potreban je certifikat (po ID-u), podaci koje treba šifrovati (bilo kao literalan string ili kao promenljiva koja može da se ograniči na string), i neobavezno, lozinka koju treba primeniti za dalje šifrovanje podataka. Sintaksa izgleda skoro isto kao za `EncryptByAsymKey`:

```
EncryptByCert(<Certificate ID>, {'<string to be encrypted>'|<string variable>}
    [, '<password>'])
```

I opet, ako se koristi lozinka da bi se podaci šifrovali ista će biti potrebna da bi se pravilno dešifrovali.

EncryptByKey

Ova funkcija ne samo da očekuje simetričan ključ (umesto drugih vrsta ključeva), već da takođe očekuje da taj ključ već bude „otvoren" (pomoću komande `OPEN SYMMETRIC KEY`) i GUID na raspolaganju za referenciranje tog ključa. Osim toga, koristi se sasvim slično, sa podacima koje treba šifrovati (bilo kao literalan string ili kao promenljiva koja može da se ograniči na string) ubačenim kao parameter i, u ovom slučaju, heš ključ neobavezno prihvaćen za proveru verodostojnosti:

```
EncryptByKey({<Key GUID>, '<string to be encrypted>'|<string variable>},
    [<add authenticator value>, '<authentication hash>'|<string variable>])
```

Pazite da, ako dodajete vrednost za proveru autentičnosti sabiranjem (u formatu `int`), ta vrednost će morati da se podnese kada se string bude dešifrovao, a morate takođe da obezbedite heš vrednost (koja će takođe biti potrebna prilikom dešifrovanja).

EncryptByPassPhrase

Ova funkcija šifruje podatke ne pomoću zvaničnog ključa, već pomoću višedelne lozinke (passphrase). Osim što prihvata višedelnu lozinku kao parameter umesto da očekuje otvoren ključ, `EncryptByPassPhrase` radi skoro isto kao `EncryptByKey`:

```
EncryptByPassPhrase({'<passphrase>'|<string variable>},
    {'<string to be encrypted>'|<string variable>},
    [<add authenticator value>, '<authentication hash>'|<string variable>])
```

Isto kao za `EncryptByKey`, ako dodajete vrednost za proveru autentičnosti sabiranjem (u formatu `int`), ta vrednost će morati da se podnese kada se string bude dešifrovao, a morate takođe da obezbedite heš vrednost.

Key_GUID

Pribavlja GUID za dati simetričan ključ u tekućoj bazi podataka:

```
Key_GUID('<Key Name>')
```

Key_ID

Pribavlja ID za dati simetričan ključ u tekućoj bazi podataka:

```
Key_ID('<Key Name>')
```

SignByAsymKey

Dodaje potpis asimetričnim ključem na datu vrednost u otvorenom tekstu:

```
SignByAsymKey(<Asymmetric Key ID>, <string variable> [, '<password>'])
```

SignByCert

Na osnovu certifikata i vrednosti u otvorenom tekstu vraća promenljivu tipa `varbinary(8000)` koja sadrži rezultujući potpis :

```
SignByCert(<Certificate ID>, <string variable> [, '<password>'])
```

VerifySignedByAsymKey

Vraća `int` (mada mi se to lično čini čudnim, pošto funkcioniše kao bit) koji ukazuje na uspeh ili neuspeh provere valjanosti potpisa za dati asimetričan ključ i vrednost u otvorenom tekstu:

```
VerifySignedByAsymKey(<Asymmetric Key ID>, <plain text> , <signature>)
```

VerifySignedByCert

Vraća `int` (i opet, to mi je čudno pošto funkcioniše kao bit) koji ukazuje na uspeh ili neuspeh provere valjanosti potpisa za dati certifikat i vrednost u otvorenom tekstu:

```
VerifySignedByCert(<Certificate ID>, <signed plain text> , <signature>)
```

KURSORSKE FUNKCIJE

Ove funkcije pribavljaju informacije o statusu ili prirodi datog kursora.

@@CURSOR_ROWS

Vraća koliko je redova trenutno u poslednjem otvorenom kursorskom skupu na trenutnoj konekciji. Obratite pažnju na to da je ovde reč o kursorima, a ne o privremenim tabelama.

Ne zaboravite da se ovaj broj resetuje svaki put kada otvorite nov kursor. Ako morate u isto vreme da otvorite više kursora, a potrebno vam je da znate broj redova u prvom kursoru, moraćete da prenesete ovu vrednost u promenljivu za čuvanje pre nego što otvorite sledeće kursore.

Ovo možete da upotrebite za kontrolu brojača u petlji `WHILE` kada radite sa kursorima, ali ja se oštro protivim takvoj praksi – vrednost u `@@CURSOR_ROWS` može da se promeni zavisno od vrste kursora i od toga da li SQL Server asinhrono popunjava kursor. Biće daleko pouzdanije upotrebiti `@@FETCH_STATUS`, a ništa manje lako za upotrebu.

Ako je vraćena vrednost negativan broj veći od –1, mora da radite sa asinhronim kursorom, pa je taj negativan broj broj zapisa do sada napravljenih u kursoru. Međutim, ako je vrednost jednaka `-1`, kursor je dinamički kursor, u kojem se broj redova stalno menja. Vraćena vrednost `0` znači ili da se nije otvarao nijedan kursor, ili da poslednji otvoreni kursor više nije otvoren. Na kraju, svaki pozitivan broj znači broj redova u kursoru.

NAPOMENA *Da biste napravili asinhroni kursor, postavite* `sp_configure cursor threshold` *na vrednost veću od* `0`*. Zatim, kada kursor pređe tu vrednost, kursor se vraća, dok se preostali zapisi asinhrono stavljaju u kursor.*

@@FETCH_STATUS

Vraća indikator statusa poslednje kursorske operacije `FETCH`.

Ako koristite kursore, koristićete `@@FETCH_STATUS`. Pomoću ove funkcije saznajete za uspeh ili neuspeh vašeg pokušaja da stignete do nekog zapisa u vašem kursoru. Vraća se konstanta koja zavisi od toga da li je SQL Server uspeo da izvrši vašu poslednju operaciju `FETCH`, a ako `FETCH` nije uspeo, zašto. Konstante su:

- `0`: Uspeh.
- `-1`: Neuspeh. Obično zato što tražite ispred početka ili iza kraja kursorskog skupa.
- `-2`: Neuspeh. Red koji ste tražili nije pronađen, obično zato što je izbrisan u međuvremenu od kad je kursorski skup napravljen pa do vašeg traženja tekućeg reda. To sme da se dogodi jedino u pomerljivim (scrollable), nedinamičkim kursorima.

Radi bolje čitljivosti, ja često pre korišćenja `@@FETCH_STATUS` postavim neke konstante.

Na primer:

```
DECLARE @NOTFOUND int
DECLARE @BEGINEND int

SELECT @NOTFOUND = -2
SELECT @BEGINEND = -1
```

Zatim u uslovima naredbe `WHILE` moje kursorske petlje mogu da koristim njih a ne sam brojač redova. Tako će kôd biti mnogo razumljiviji.

CURSOR_STATUS

Funkcija `CURSOR_STATUS` omogućava pozivaocu snimljene procedure da utvrdi da li je ta procedura vratila kursorski i rezultujući skup. Sintaksa je sledeća:

```
CURSOR_STATUS
   (
      {'<local>', '<cursor name>'}
      | {'<global'>, '<cursor name>'}
      | {'<variable>', '<cursor variable>'}
   )
```

`local`, `global` i `variable` su konstante koje označavaju izvor kursora. `local` je jednako imenu lokalnog kursora, `global` imenu globalnog kursora, a `variable` lokalne promenljive.

Ako koristite oblik `cursor name`, moguće su četiri vraćene vrednosti:

- `1`: Kursor je otvoren. Ako je kursor dinamičan, njegov rezultujući skup ima nula ili više redova. Ako kursor nije dinamičan, on ima jedan red ili više njih.
- `0`: Rezultujući skup kursora je prazan.
- `-1`: Kursor je zatvoren.
- `-3`: Kursor po imenu `cursor name` ne postoji.

Ako koristite oblik `cursor variable` , moguće je pet vraćenih vrednosti:

- `1`: Kursor je otvoren. Ako je kursor dinamičan, njegov rezultujući skup ima nula ili više redova. Ako kursor nije dinamičan, on ima jedan red ili više njih.
- `0`: Rezultujući skup kursora je prazan.
- `-1`: Kursor je zatvoren.
- `-2`: Nijedan kursor nije dodeljen promenljivoj `cursor variable`.
- `-3`: Promenljiva po imenu `cursor name` ne postoji, ili ako postoji, još joj nije alociran kursor.

FUNKCIJE ZA DATUM I VREME

Ovo je područje sa nekoliko elemenata uvedenih u SQL Serveru 2008. Pored rada nad podacima vremenskih pečata (što je u suštini pre orijentisano na verzije nego na časovnike ili kalendare), datumske i vremenske funkcije vrše operacije nad vrednostima bilo kojeg od različitih datumskih i vremenskih tipova podataka koje podržava SQL Server.

U radu sa mnogim od tih funkcija, SQL Server prepoznaje jedanaest „delova datuma" (datepart) i njihove skraćenice, prikazane u tabeli B-3.

TABELA B-3: Delovi datuma i njihove skraćenice

DEO DATUMA (DATEPART)	SKRAĆENICE
godina (Year)	yy, yyyy
kvartal (Quarter)	qq, q
mesec (Month)	mm, m
dan u godini (Dayofyear)	dy, y
dan (Day)	dd, d
nedelja (Week)	wk, ww
dan u nedelji (Weekday)	dw
sat (Hour)	hh
minut (Minute)	mi, n

DEO DATUMA (DATEPART)	SKRAĆENICE
sekunda (Second)	ss, s
milisekunda (Millisecond)	ms

CURRENT_TIMESTAMP

Funkcija CURRENT_TIMESTAMP jednostavno vraća tekući datum i vreme kao tip datetime. Ona je ekvivalentna funkciji GETDATE(). Sintaksa je sledeća:

```
CURRENT_TIMESTAMP
```

DATEADD

Funkcija DATEADD dodaje interval na datum i vraća novi datum. Sintaksa je sledeća:

```
DATEADD(<datepart>, <number>, <date>)
```

Argument *datepart* određuje vremensku skalu intervala (dan, nedelja, mesec i tako dalje) i može biti bilo koji od delova datuma koje SQL Server prepoznaje. Argument number je broj takvih delova datuma koje treba dodati na datum date.

DATEDIFF

Funkcija DATEDIFF vraća razliku između dva navedena datuma u zahtevanim jedinicama vremena (na primer: sati, dani, nedelje). Sintaksa je sledeća:

```
DATEDIFF(<datepart>, <startdate>, <enddate>)
```

Argument datepart može biti bilo koji od delova datuma koje SQL Server prepoznaje i određuje zahtevanu jedinicu vremena u kojoj treba prikazati razliku između datuma *start-date* i datuma *enddate*.

DATETIMEFROMPARTS

Svaki od tipova podataka za datum i vreme možete da sastavite od delova. Verzije ove funkcije i njihovi vraćeni tipovi mogu da budu:

- **DATEFROMPARTS** vraća DATE
- **DATETIMEFROMPARTS** vraća DATETIME
- **DATETIME2FROMPARTS** vraća DATETIME2
- **DATETIMEOFFSETFROMPARTS** vraća DATETIMEOFFSET
- **SMALLDATETIMEFROMPARTS** vraća SMALLDATETIME
- **TIMEFROMPARTS** vraća TIME

Svaka od ovih funkcija sastavlja svoj izlaz nekom kombinacijom sledećih delova, koji se predaju kao argumenti.

- **year**: Celobrojni izraz koji znači godinu.
- **month**: Celobrojni izraz koji znači mesec.
- **day**: Celobrojni izraz koji znači dan.
- **hour**: Celobrojni izraz koji znači sate.
- **minute**: Celobrojni izraz koji znači minute.
- **seconds**: Celobrojni izraz koji znači sekunde.
- **fractions**: Celobrojni izraz koji znači delove (`milliseconds` za `DATETIMEFROMPARTS`).
- **hour_offset**: Celobrojni izraz koji znači deo pomeraja vremenske zone koji se odnosi na časove.
- **minute_offset**: Celobrojni izraz koji znači deo pomeraja vremenske zone koji se odnosi na minute.
- **precision**: Celobrojni literal koji određuje zahtevanu preciznost vrednosti `datetimeoffset` koja treba da se vrati. Ako se izostavi argument preciznosti, primeniće se podrazumevana preciznost 7.

Većina ovih argumenata je jasna sama po sebi, ali je argument `fractions` jedinstven. `Fractions` su delovi sekunde, a veličina jednog fragmenta zavisi od prosleđenog parametra `precision`. `TIME` sa `precision` jednakom 7 ima granularnost od 100 nanosekundi (sedmo decimalno mesto), pa su u tom kontekstu inkrementi `fractions` od po 100 nanosekundi. Ako je `precision` jednako 3 imamo milisekunde (tri decimale), pa su `fractions` milisekunde. Ali, ako navedete `precision` jednako 0, morate da stavite i `fractions` jednako 0 ili se javlja greška.

Kao što vidite, sintakse svih verzija ove funkcije su slične:

```
DATEFROMPARTS ( year, month, day )

DATETIMEFROMPARTS ( year, month, day, hour, minute, seconds, milliseconds )

DATETIME2FROMPARTS ( year, month, day, hour, minute, seconds, fractions,
precision )

DATETIMEOFFSETFROMPARTS ( year, month, day, hour, minute, seconds,
fractions,

hour_offset, minute_offset, precision )

SMALLDATETIMEFROMPARTS ( year, month, day, hour, minute )

TIMEFROMPARTS ( hour, minute, seconds, fractions, precision )
```

DATENAME

Funkcija `DATENAME` vraća string koji predstavlja naziv (na primer: 1999, Thursday, ili July) navedenog dela (`datepart`) navedenog datuma (`date`). Sintaksa je sledeća:

```
DATENAME(<datepart>, <date>)
```

DATEPART

Funkcija `DATEPART` vraća ceo broj koji predstavlja navedeni deo (`datepart`) navedenog datuma (`date`). Sintaksa je sledeća:

```
DATEPART(<datepart>, <date>)
```

Funkcija `DAY` je ekvivalentna `DATEPART(dd, <date>)`; `MONTH` je ekvivalentna `DATEPART(mm, <date>)`; `YEAR` je ekvivalentna `DATEPART(yy, <date>)`.

DAY

Funkcija `DAY` vraća ceo broj koji predstavlja deo navedenog datuma koji se odnosi na dan. Sintaksa je sledeća:

```
DAY(<date>)
```

Funkcija `DAY` je ekvivalentna sa `DATEPART(dd, <date>)`.

EOMONTH

`EOMONTH` vraća poslednji datum u mesecu koji zatekne u svom parametru `date`. Izgleda ovako:

```
EOMONTH( date_value )
```

GETDATE

Funkcija `GETDATE` vraća tekući sistemski datum i vreme. Sintaksa je sledeća:

```
GETDATE()
```

GETUTCDATE

Funkcija `GETUTCDATE` vraća tekuće UTC (Universal Time Coordinate) vreme. Drugim rečima, vraća se GMT (Greenwich Mean Time). Vrednost se izvodi tako što se od servera uzme lokalno vreme i lokalna vremenska zona, pa se od toga izračuna GMT. Uključuje se i letnje računanje vremena. `GETUTCDATE` ne može da se poziva iz korisnički definisane funkcije. Sintaksa je sledeća:

```
GETUTCDATE()
```

ISDATE

Funkcija `ISDATE` utvrđuje da li je podneti izraz ispravan datum. Sintaksa je sledeća:

```
ISDATE(<expression>)
```

MONTH

Funkcija `MONTH` vraća ceo broj koji predstavlja deo navedenog datuma koji se odnosi na mesec. Sintaksa je sledeća:

```
MONTH(<date>)
```

Funkcija `MONTH` je ekvivalentna sa `DATEPART(mm, <date>)`.

SYSDATETIME

Veoma slično starodrevnoj funkciji `GETDATE`, `SYSDATETIME` vraća tekući sistemski datum i vreme. Razlike su dve: prva, `SYSDATETIME` vraća veći nivo preciznosti. Druga, novija funkcija vraća noviji tip podatka `datetime2` (da bi se podržala veća preciznost – u ovom slučaju, preciznost 7). Sintaksa je sledeća:

```
SYSDATETIME()
```

SYSDATETIMEOFFSET

Slično `SYSDATETIME`, ova funkcija vraća tekući sistemski datum i vreme. Međutim, `SYSDATETIMEOFFSET` umesto jednostavnog tipa podatka `datetime2`, vraća vreme u novom tipu podatka `datetimeoffset` (sa preciznošću 7), pa se dobija informacija o pomeraju u odnosu na univerzalno vreme. Sintaksa je sledeća:

```
SYSDATETIMEOFFSET()
```

SYSUTCDATETIME

Veoma slično starodrevnoj funkciji `GETUTCDATE`, `SYSUTCDATETIME` vraća tekući UTC datum i vreme. `SYSUTCDATETIME`, međutim, vraća noviji tip podatka `datetime2` (sa preciznošću 7). Sintaksa je sledeća:

```
SYSUTCDATETIME()
```

SWITCHOFFSET

Vraća vrednost `datetimeoffset` promenjenu iz pomeraja sačuvane vremenske zone u pomeraj navedene nove vremenske zone.

Dakle, na primer, možete da vidite kako se pomeraj primenjuje na jednom primeru:

```
   SWITCHOFFSET ( <datetime to offset>, <time zone offset amount> )
CREATE TABLE dbo.test
   (
   ColDatetimeoffset datetimeoffset
   );
GO
```

```
INSERT INTO dbo.test
VALUES ('1998-09-20 7:45:50.71345 -5:00');
GO
SELECT SWITCHOFFSET (ColDatetimeoffset, '-08:00')
FROM dbo.test;
GO
--Returns: 1998-09-20 04:45:50.7134500 -08:00
SELECT ColDatetimeoffset
FROM dbo.test;
--Returns: 1998-09-20 07:45:50.7134500 -05:00
```

TODATETIMEOFFSET

Prihvata dati komad datumsko/vremenske informacije i dodaje zadati vremenski pomeraj, pa proizvodi tip podatka `datetimeoffset`. Sintaksa je:

```
TODATETIMEOFFSET(<data that resolves to datetime>, <time zone>)
```

Pa, na primer:

```
DECLARE @OurDateTimeTest datetime;
SELECT @OurDateTimeTest = '2008-01-01 12:54';
SELECT TODATETIMEOFFSET(@OurDateTimeTest, '-07:00');
```

daje:

```
-----------------------------------
1/1/2008 12:54:00 PM -07:00

(local)(sa): (1 row(s) affected)
```

YEAR

Funkcija `YEAR` vraća ceo broj koji predstavlja deo navedenog datuma koji se odnosi na godinu. Sintaksa je sledeća:

```
YEAR(<date>)
```

Funkcija `YEAR` je ekvivalentna sa `DATEPART(yy, <date>)`.

HIJERARHIJSKE FUNKCIJE

Kao što je već pomenuto, nova implementacija hijerarhijskih ID-a donekle prevazilazi namene ove knjige. Za detaljna objašnjenja o korišćenju sledećih funkcija, molim vas da potražite referencu „HierarchyId data type method" u Books Online. U ovom odeljku su jednostavno nabrojane i ukratko opisane funkcije SQL Servera 2012 za pomoć u radu sa hijerarhijskim ID-ima i stablima vezanim za njih.

GetAncestor

Funkcija `GetAncestor` vraća hierarchyID za ntog pretka elementa u pitanju. Sintaksa je sledeća:

```
GetAncestor(<numeric expression>)
```

Predatim parametrom određujete sa kojeg nivoa u lancu hijerarhije tražite potomke. Na primer, 1 vraća decu elementa u pitanju, 2 vraća unuke, a 0 vraća hierarchyID samog elementa u pitanju.

GetDescendant

Vraća čvor dete tog roditelja.

GetLevel

Vraća ceo broj koji predstavlja dubinu čvora u stablu.

GetRoot

Vraća koren hijerarhijskog stabla.

IsDescendantOf

Vraća `true` ako je dete potomak elementa u pitanju.

Parse

Konvertuje kanonsku string predstavu hierarchyID u vrednost tipa hierarchyID. Parse se poziva implicitno kada se dogodi konverzija iz tipa string u hierarchyID.

GetReparentedValue

Vraća čvor čija putanja od korena je jednaka putanji do `newRoot`, kojoj sledi putanja od `oldRoot` do tekućeg elementa.

ToString

Funkcionalni ekvivalent korišćenju funkcije `CAST`, za hijerarhijski čvor na koji se primeni funkcija vraća se vrednost tipa string.

MATEMATIČKE FUNKCIJE

Matematičke funkcije vrše izračunavanja. To su:

- `ABS`
- `ACOS`

- ASIN
- ATAN
- ATN2
- CEILING
- COS
- COT
- DEGREES
- EXP
- FLOOR
- LOG
- LOG10
- PI
- POWER
- RADIANS
- RAND
- ROUND
- SIGN
- SIN
- SQRT
- SQUARE
- TAN

ABS

Funkcija `ABS` vraća pozitivnu, apsolutnu vrednost `numeričkog izraza`. Sintaksa je sledeća:

```
ABS(<numeric expression>)
```

ACOS

Funkcija `ACOS` vraća ugao u radijanima čiji kosinus je jednak izrazu `expression` (drugim rečima, vraća arkus kosinus izraza `expression`). Sintaksa je sledeća:

```
ACOS(<expression>)
```

Vrednost `expression` mora biti između `-1` i `1` i mora da bude tipa podatka `float`.

ASIN

Funkcija `ASIN` vraća ugao u radijanima čiji sinus je jednak `expression` (drugim rečima, vraća arkus sinus izraza `expression`). Sintaksa je sledeća:

```
ASIN(<expression>)
```

Vrednost izraza mora biti između `-1` i `1` i mora da bude tipa podatka `float`.

ATAN

Funkcija `ATAN` vraća ugao u radijanima čiji tangens je jednak `expression` (drugim rečima, vraća arkus tangens `expression`). Sintaksa je sledeća:

```
ATAN(<expression>)
```

`Expression` mora da bude tipa podatka `float`.

ATN2

Funkcija `ATN2` vraća ugao u radijanima čiji tangens je između dva podneta izraza (drugim rečima, vraća arkus tangens dva izraza). Sintaksa je sledeća:

```
ATN2(<expression1>, <expression2>)
```

Izrazi `expression1` i `expression2` moraju da budu tipa podatka `float`.

CEILING

Funkcija `CEILING` vraća najmanji ceo broj jednak ili veći od navedenog izraza. Sintaksa je sledeća:

```
CEILING(<expression>)
```

COS

Funkcija `COS` vraća kosinus ugla navedenog kao `expression`. Sintaksa je sledeća:

```
COS(<expression>)
```

Podneti ugao mora biti u radijanima i `expression` mora da bude tipa podatka `float`.

COT

Funkcija `COT` vraća kotangens ugla navedenog kao `expression`. Sintaksa je sledeća:

```
COT(<expression>)
```

Podneti ugao mora biti u radijanima i `expression` mora da bude tipa podatka `float`.

DEGREES

Funkcija `DEGREES` prihvata ugao u radijanima (`expression`) a vraća ugao u stepenima. Sintaksa je sledeća:

```
DEGREES(<expression>)
```

EXP

Funkcija `EXP` vraća eksponencijalnu vrednost vrednosti date kao `expression`. Sintaksa je sledeća:

```
EXP(<expression>)
```

`Expression` mora da bude tipa podatka `float`.

FLOOR

Funkcija `FLOOR` vraća najveći ceo broj jednak ili manji od vrednosti date kao `expression`. Sintaksa je sledeća:

```
FLOOR(<expression>)
```

LOG

Funkcija `LOG` vraća prirodni logaritam vrednosti navedene kao `expression`. Sintaksa je sledeća:

```
LOG(<expression>)
```

`Expression` mora da bude tipa podatka `float`.

LOG10

Funkcija `LOG10` vraća logaritam sa osnovom 10 za vrednost navedenu kao `expression`. Sintaksa je sledeća:

```
LOG10(<expression>)
```

`Expression` mora da bude tipa podatka `float`.

PI

Funkcija `PI` vraća vrednost te konstante. Sintaksa je sledeća:

```
PI()
```

POWER

Funkcija `POWER` podiže vrednost navedenu kao `expression` na navedeni stepen `power`. Sintaksa je sledeća:

```
POWER(<expression>, <power>)
```

RADIANS

Funkcija `RADIANS` vraća ugao u radijanima koji odgovara uglu u stepenima navedenom kao `expression`. Sintaksa je sledeća:

```
RADIANS(<expression>)
```

RAND

Funkcija `RAND` vraća slučajno izabranu vrednost između `0` i `1`. Sintaksa je sledeća:

```
RAND([<seed>])
```

Neobavezna vrednost semena `seed` je celobrojni izraz, koji određuje početnu vrednost za generisanje slučajnog broja. Možete da pustite SQL Server da generiše vlastitu vrednost semena kad god se funkcija pozove, ili možete sami da odredite seme.

NAPOMENA *Budite obazrivi ako koristite literal kao vrednost semena. Za konkretnu vrednost semena, SQL Server uvek vraća isti rezultat (što znači da vaš* `RAND` *naglo prestaje da bude slučajan). Ako vam je zaista potrebno generisanje slučajnog broja, možete da pustite SQL Server da izabere slučajno seme, ili da upotrebite seme koje se stalno menja, kao što je broj nanosekundi od konkretne tačke u vremenu.*

ROUND

Funkcija `ROUND` prihvata broj naveden kao `expression` i zaokružava ga na zadatu dužinu:

```
ROUND(<expression>, <length> [, <function>])
```

Parametar `length` određuje preciznost na koju `expression` treba da se zaokruži. Parametar `length` mora biti tipa podatka `tinyint`, `smallint`, ili `int`. Neobavezan parametar `function` može da se upotrebi da bi se navelo da li broj treba da se zaokruži ili odseče. Ako se izostavi vrednost `function` ili je ona jednaka `0` (podrazumevano), vrednost `expression` će se zaokružiti. Ako se upotrebi vrednost različita od `0`, vrednost `expression` biće odsečena.

SIGN

Funkcija `SIGN` vraća predznak `expression`. Moguće su vraćene vrednosti `+1` za pozitivan broj, `0` za nulu, a `–1` za negativan broj. Sintaksa je sledeća:

```
SIGN(<expression>)
```

SIN

Funkcija `SIN` vraća sinus ugla. Sintaksa je sledeća:

```
SIN(<angle>)
```

Ugao `angle` mora biti u radijanima i mora da bude tipa podatka `float`. Vraćena vrednost će takođe biti tipa podatka `float`.

SQRT

Funkcija `SQRT` vraća kvadratni koren vrednosti date kao `expression`. Sintaksa je sledeća:

```
SQRT(<expression>)
```

`Expression` mora da bude tipa podatka `float`.

SQUARE

Funkcija `SQUARE` vraća kvadrat vrednosti podnete kao `expression`. Sintaksa je sledeća:

```
SQUARE(<expression>)
```

`Expression` mora da bude tipa podatka `float`.

TAN

Funkcija `TAN` vraća tangens vrednosti predate kao `expression`. Sintaksa je sledeća:

```
TAN(<expression>)
```

Parametar `expression` navodi broj radijana i mora da bude tipa podatka `float` ili `real`.

OSNOVNE FUNKCIJE METAPODATAKA

Funkcije metapodataka pribavljaju informacije o bazi podataka i objektima baze podataka. To su:

- `COL_LENGTH`
- `COL_NAME`
- `COLUMNPROPERTY`

- DATABASEPROPERTY
- DATABASEPROPERTYEX
- DB_ID
- DB_NAME
- FILE_ID
- FILE_NAME
- FILEGROUP_ID
- FILEGROUP_NAME
- FILEGROUPPROPERTY
- FILEPROPERTY
- FULLTEXTCATALOGPROPERTY
- FULLTEXTSERVICEPROPERTY
- INDEX_COL
- INDEXKEY_PROPERTY
- INDEXPROPERTY
- OBJECT_ID
- OBJECT_NAME
- OBJECTPROPERTY
- OBJECTPROPERTYEX
- @@PROCID
- SCHEMA_ID
- SCHEMA_NAME
- SQL_VARIANT_PROPERTY
- TYPE_ID
- TYPE_NAME
- TYPEPROPERTY

COL_LENGTH

Funkcija `COL_LENGTH` vraća definisanu dužinu kolone. Sintaksa je sledeća:

```
COL_LENGTH('<table>', '<column>')
```

Parametar `column` je ime kolone čija dužina se traži. Parametar `table` je ime tabele koja sadrži tu kolonu.

COL_NAME

Funkcija `COL_NAME` prihvata ID broj tabele i ID broj kolone a vraća ime kolone u bazi podataka. Sintaksa je sledeća:

```
COL_NAME(<table_id>, <column_id>)
```

Parametar `column_id` je ID broj kolone. Parametar `table_id` je ID broj tabele koja sadrži tu kolonu.

COLUMNPROPERTY

Funkcija `COLUMNPROPERTY` vraća podatak o koloni ili proceduri podnetoj kao parametar. Sintaksa je sledeća:

```
COLUMNPROPERTY(<id>, <column>, <property>)
```

Parametar `id` je ID tabele/procedure. Parametar `column` je ime parametra kolone. Parametar `property` određuje podatak koji treba da se vrati za parametar kolone ili procedure. Parametar `property` može imati jednu od sledećih vrednosti:

- `AllowsNull`: dozvoljava `NULL` vrednosti.
- `IsComputed`: kolona je izračunata kolona.
- `IsCursorType`: procedura je tipa `CURSOR`.
- `IsFullTextIndexed`: kolona indeksirana potpunim tekstom.
- `IsIdentity`: kolona je `IDENTITY` kolona.
- `IsIdNotForRepl`: kolona proverava `IDENTITY NOT FOR REPLICATION`.
- `IsOutParam`: parametar procedure je izlazni parametar.
- `IsRowGuidCol`: kolona je `ROWGUIDCOL` kolona.
- `Precision`: preciznost za tip podatka u parametru kolone.
- `Scale`: broj decimala za tip podatka u parametru kolone.
- `UseAnsiTrim`: ANSI parametar popunjavanja je bio `ON` kada se pravila tabela.

Vrednost vraćena od ove funkcije će biti `1` za tačan (`True`), `0` za netačan (`False`), a `NULL` ako je ulaz bio neispravan – osim za `Precision` (kada funkcija vraća preciznost tipa podatka) i `Scale` (kada vraća broj decimala).

DATABASEPROPERTY

Funkcija `DATABASEPROPERTY` vraća podešavanje za navedenu bazu podataka i ime svojstva. Sintaksa je sledeća:

```
DATABASEPROPERTY('<database>', '<property>')
```

Parametar `database` je ime baze podataka za koju se podatak navedenog svojstva vraća. Parametar `property` sadrži ime svojstva baze podataka, a to može da bude jedna od vrednosti iz tabele B-4.

TABELA B-4: Vrednosti svojstava baze podataka

VREDNOST	OPIS
`IsAnsiNullDefault`	Baza podataka primenjuje standard ANSI-92 za `NULL` vrednosti.
`IsAnsiNullsEnabled`	Poređenja sa `NULL` nisu omogućana.
`IsAnsiWarningsEnabled`	Poruke upozorenja sa izdaju za standardna stanja greške.
`IsAutoClose`	Baza podataka oslobađa resurse kada je napusti poslednji korisnik.
`IsAutoShrink`	Datoteke baze podataka mogu automatski i povremeno da se skraćuju.
`IsAutoUpdateStatistics`	Omogućena je opcija automatskog ažuriranja statistike.
`IsBulkCopy`	Baza podataka dozvoljava operacije bez prijavljivanja (kao što je Bulk Copy Program).
`IsCloseCursorsOnCommitEnabled`	Prilikom potvrđivanja transakcije zatvaraju se svi otvoreni kursori.
`IsDboOnly`	Baza podataka je dostupna jedino vlasniku (`dbo`).
`IsDetached`	Sa bazom podataka je veza raskinuta operacijom `detach`.
`IsEmergencyMode`	Baza podataka je u režimu vanrednog stanja.
`IsFulltextEnabled`	U bazi podataka je omogućeno korišćenje potpunog teksta.
`IsInLoad`	Baza podataka se učitava.
`IsInRecovery`	Baza podataka se oporavlja.
`IsInStandby`	Baza podataka je u režimu samo za čitanje i dozvoljeno je evidentiranje obnavljanja.
`IsLocalCursorsDefault`	Deklaracije kursora su podrazumevano `LOCAL`.
`IsNotRecovered`	Baza podataka nije uspela da se oporavi.
`IsNullConcat`	Kada se `NULL` spaja sa stringom vraća se `NULL`.
`IsOffline`	Baza podataka je oflajn.

VREDNOST	OPIS
IsQuotedIdentifiersEnabled	Identifikatori mogu da se ograniče dvostrukim navodnicima.
IsReadOnly	Baza podataka je u režimu samo za čitanje.
IsRecursiveTriggersEnabled	Omogućeno je rekurzivno ispaljivanje okidača.
IsShutDown	Baza podataka je naišla na problem tokom pokretanja.
IsSingleUser	Baza podataka je u režimu za jednog korisnika.
IsSuspect	Baza podataka je sumnjiva.
IsTruncLog	Baza podataka odseca kontrolne tačke prijavljivanja.
Version	Interni broj verzije SQL Server koda kojim je baza podataka napravljena.

Vraćena vrednost će biti `1` za tačan, `0` za netačan, a `NULL` ako je podneta neispravna vrednost – osim za `Version` (kada funkcija vraća broj verzije ako je baza podataka otvorena, a `NULL` ako je baza podataka zatvorena).

DATABASEPROPERTYEX

Funkcija `DATABASEPROPERTYEX` je u suštini nadskup od `DATABASEPROPERTY`, jer takođe vraća podešavanje za navedenu bazu podataka i ime svojstva. Sintaksa je sasvim slična kao za `DATABASEPROPERTY` i glasi:

```
DATABASEPROPERTYEX('<database>', '<property>')
```

`DATABASEPROPERTYEX` jedino ima na raspolaganju još nekoliko svojstava, kao što su:

- `Collation`: Vraća podrazumevani redosled sravnjivanja za bazu podataka (ne zaboravite da redosled za sravnjivanje može da se nadjača na nivou kolone).
- `ComparisonStyle`: Ukazuje na Windowsov stil poređenja (na primer, razlikovanje velikih i malih slova) konkretnog redosleda sravnjivanja.
- `IsAnsiPaddingEnabled`: Da li se stringovi popunjavaju do jednake dužine pre poređenja ili insertovanja.
- `IsArithmaticAbortEnabled`: Da li se upiti prekidaju ako dođe do greške prekoračenja podataka ili deljenja nulom.

Parametar `database` je ime baze podataka za koju se vraća podatak navedenog svojstva. Parametar `property` je ime svojstva baze podataka i može biti jedna od predviđenih vrednosti.

DB_ID

Funkcija `DB_ID` vraća ID broj baze podataka. Sintaksa je sledeća:

```
DB_ID(['<database name>'])
```

Neobavezan parametar `database_name` navodi za koju bazu podataka se traži ID broj. Ako se izostavi parametar `database_name`, vratiće se ID broj tekuće baze podataka.

DB_NAME

Funkcija `DB_NAME` vraća ime baze podataka koja nosi navedeni ID broj. Sintaksa je sledeća:

```
DB_NAME([<database_id>])
```

Neobavezan parametar `database_id` navodi za koju bazu podataka se traži ime. Ako se izostavi parametar `database_id`, vratiće se ime tekuće baze podataka.

FILE_ID

Funkcija `FILE_ID` vraća ID broj datoteke za navedeno ime datoteke u tekućoj bazi podataka. Sintaksa je sledeća:

```
FILE_ID('<file_name>')
```

Parametar `file_name` znači ime datoteke za koju se traži ID.

FILE_NAME

Funkcija `FILE_NAME` vraća ime datoteke za navedeni ID broj. Sintaksa je sledeća:

```
FILE_NAME(<file_id>)
```

Parametar `file_id` znači ID broj datoteke za koju se traži ime.

FILEGROUP_ID

Funkcija `FILEGROUP_ID` vraća ID broj grupe datoteka za navedeno ime grupe datoteka. Sintaksa je sledeća:

```
FILEGROUP_ID('<filegroup_name>')
```

Parametar `filegroup_name` znači ime grupe datoteka za koju se traži ID broj grupe datoteka.

FILEGROUP_NAME

Funkcija `FILEGROUP_NAME` vraća ime grupe datoteka za navedeni ID broj grupe datoteka. Sintaksa je sledeća:

```
FILEGROUP_NAME(<filegroup_id>)
```

Parametar `filegroup_id` znači ID broj grupe datoteka za koju se traži ime.

FILEGROUPPROPERTY

Funkcija `FILEGROUPPROPERTY` vraća podešavanje navedenog svojstva grupe datoteka, ako je dato ime grupe datoteka i ime svojstva. Sintaksa je sledeća:

```
FILEGROUPPROPERTY(<filegroup_name>, <property>)
```

Parametar `filegroup_name` znači ime grupe datoteka čije svojstvo se ispituje. Parametar `property` znači svojstvo koje se ispituje, a može da bude jedna od sledećih vrednosti:

- `IsReadOnly`: grupa datoteka je samo za čitanje.
- `IsUserDefinedFG`: grupa datoteka je korisnički definisana grupa datoteka.
- `IsDefault`: grupa datoteka je podrazumevana grupa datoteka.

Vraćena vrednost će biti `1` za tačan (`True`), `0` za netačan (`False`), a `NULL` ako je podneta neispravna vrednost.

FILEPROPERTY

Funkcija `FILEPROPERTY` vraća podešavanje navedenog svojstva datoteke, ako je dato ime datoteke i ime svojstva. Sintaksa je sledeća:

```
FILEPROPERTY(<file_name>, <property>)
```

Parametar `file_name` znači ime datoteke koja sadrži svojstvo koje se ispituje. Parametar `property` znači svojstvo koje se ispituje, a može da bude jedna od sledećih vrednosti:

- `IsReadOnly`: datoteka je samo za čitanje.
- `IsPrimaryFile`: datoteka je osnovna datoteka.
- `IsLogFile`: datoteka je datoteka dnevnika.
- `SpaceUsed`: količina prostora koji zauzima navedena datoteka.

Vraćena vrednost će biti `1` za tačan (`True`), `0` za netačan (`False`), a `NULL` ako je podneta neispravna vrednost, osim za `SpaceUsed` (kada se vraća broj alociranih stranica u datoteci).

FULLTEXTCATALOGPROPERTY

Funkcija `FULLTEXTCATALOGPROPERTY` vraća podatke o svojstvima kataloga potpunog teksta. Sintaksa je sledeća:

```
FULLTEXTCATALOGPROPERTY(<catalog_name>, <property>)
```

Parametar `catalog_name` znači ime kataloga potpunog teksta. Parametar `property` znači svojstvo koje se ispituje. Mogu da se ispituju sledeća svojstva:

- `PopulateStatus`: za koje mogu da se vrate vrednosti `0` (stanje mirovanja), `1` (popunjavanje u toku), `2` (pauziran), `3` (prigušen), `4` (oporavlja se), `5` (isključen), `6` (inkrementalno popunjavanje u toku) i `7` (ažuriranje indeksa).

- `ItemCount`: vraća broj elemenata indeksiranih potpunim tekstom trenutno u katalogu potpunog teksta.
- `IndexSize`: vraća veličinu indeksa potpunog teksta u megabajtima.
- `UniqueKeyCount`: vraća broj jedinstvenih reči koje čine indeks potpunog teksta u ovom katalogu.
- `LogSize`: vraća veličinu (u bajtovima) kombinovanog skupa dnevnika grešaka udruženog sa katalogom potpunog teksta.
- `PopulateCompletionAge`: vraća razliku (u sekundama) između završetka poslednjeg popunjavanja indeksa potpunog teksta i 01/01/1990 00:00:00.

FULLTEXTSERVICEPROPERTY

Funkcija `FULLTEXTSERVICEPROPERTY` vraća podatke o svojstvima servisa potpunog teksta. Sintaksa je sledeća:

```
FULLTEXTSERVICEPROPERTY(<property>)
```

Parametar `property` je ime svojstva servisa koje se ispituje. Parametar `property` može imati jednu od sledećih vrednosti:

- `ResourceUsage`: vraća vrednost od `1` (u pozadini) do `5` (namenski).
- `ConnectTimeOut`: vraća broj sekundi koje će Search Service da čeka na sve konekcije sa SQL Serverom da bi počeo sa popunjavanjem indeksa potpunog teksta pre nego što odustane.
- `IsFulltextInstalled`: vraća `1` ako je Full-Text Service instaliran na računaru, a inače vraća `0`.

INDEX_COL

Funkcija `INDEX_COL` vraća ime indeksirane kolone. Sintaksa je sledeća:

```
INDEX_COL('<table>', <index_id>, <key_id>)
```

Parametar `table` je ime tabele, `index_id` je ID broj indeksa, a `key_id` je ID ključa.

INDEXKEY_PROPERTY

Ova funkcija vraća informacije o indeksnom ključu.

```
INDEXKEY_PROPERTY(<table_id>, <index_id>, <key_id>, <property>)
```

Parametar `table_id` je numerički ID tipa podatka `int`, koji definiše tabelu koju hoćete da pregledate. Poslužite se funkcijom `OBJECT_ID` da biste dobili numerički `table_id`. Parametar `index_id` je ID indeksa, i takođe je tipa podatka `int`. Parametar `key_id` je pozicija kolone u indeksnom ključu; na primer, ako je ključ nad tri kolone, postavljanje ove vrednosti na `2` će značiti da hoćete da pregledate srednju kolonu. Na kraju, parametar `property`

je identifikator niza znakova jednog od dva svojstva čije podešavanje vas zanima. Dve moguće vrednosti su `ColumnId`, koja vraća ID fizičke kolone i `IsDescending`, koja vraća redosled kojim je kolona sortirana (`1` za opadajući redosled, a `0` za rastući).

INDEXPROPERTY

Funkcija `INDEXPROPERTY` vraća podešavanje navedenog svojstva indeksa, za dati ID tabele, ime indeksa i ime svojstva. Sintaksa je sledeća:

```
INDEXPROPERTY(<table ID>, <index>, <property>)
```

Parametar `property` znači koje svojstvo indeksa se ispituje. Parametar `property` može imati jednu od sledećih mogućih vrednosti:

- `IndexDepth`: dubina indeksa.
- `IsAutoStatistic`: indeks je napravljen opcijom `autocreate statistics` snimljene procedure `sp_dboption`.
- `IsClustered`: indeks je klasterovan.
- `IsStatistics`: indeks je napravljen naredbom `CREATE STATISTICS` ili opcijom `auto-create statistics` snimljene procedure `sp_dboption`.
- `IsUnique`: indeks je jedinstven.
- `IndexFillFactor`: indeks određuje vlastiti procenat popunjenosti.
- `IsPadIndex`: indeks određuje prostor koji treba da ostaje otvoren u svakom unutrašnjem čvoru.
- `IsFulltextKey`: indeks je ključ potpunog teksta za neku tabelu.
- `IsHypothetical`: indeks je hipotetski i ne može da se koristi direktno kao putanja za pristup podacima.

Vraćena vrednost iz ove funkcije će biti `1` za tačan (`True`), `0` za netačan (`False`) i `NULL` ako je ulaz bio neispravan, osim za `IndexDepth` (koji vraća broj nivoa u tom indeksu) i `IndexFillFactor` (koji vraća procenat popunjavanja primenjen kada je indeks pravljen ili poslednji put ponovo građen).

OBJECT_ID

Funkcija `OBJECT_ID` vraća ID broj navedenog objekta baze podataka. Sintaksa je sledeća:

```
OBJECT_ID('<object>')
```

OBJECT_NAME

Funkcija `OBJECT_NAME` vraća ime navedenog objekta baze podataka. Sintaksa je sledeća:

```
OBJECT_NAME(<object id>)
```

OBJECTPROPERTY

Funkcija `OBJECTPROPERTY` vraća podatke o objektima u tekućoj bazi podataka. Sintaksa je sledeća:

```
OBJECTPROPERTY(<id>, <property>)
```

Parametar `id` je ID zahtevanog objekta. Parametar `property` određuje zahtevanu informaciju o objektu. Dozvoljene su sledeće vrednosti za `property` :

CnstIsClustKey	HasUpdateTrigger
CnstIsColumn	IsAnsiNullsOn
CnstIsDeleteCascade	IsCheckCnst
CnstIsDisabled	IsConstraint
CnstIsNonclustKey	IsDefault
CnstIsNotRepl	IsDefaultCnst
CnstIsNotTrusted	IsDeterministic
CnstIsUpdateCascade	IsExecuted
ExecIsAfterTrigger	IsExtendedProc
ExecIsAnsiNullsOn	IsForeignKey
ExecIsDeleteTrigger	IsIndexable
ExecIsFirstDeleteTrigger	IsIndexed
ExecIsFirstInsertTrigger	IsInlineFunction
ExecIsFirstUpdateTrigger	IsMSShipped
ExecIsInsertTrigger	IsPrimaryKey
ExecIsInsteadOfTrigger	IsProcedure
ExecIsLastDeleteTrigger	IsQueue
ExecIsLastInsertTrigger	IsQuotedIdentOn
ExecIsLastUpdateTrigger	IsReplProc
ExecIsQuotedIdentOn	IsRule
ExecIsStartup	IsScalarFunction
ExecIsTriggerDisabled	IsSchemaBound
ExecIsTriggerNotForRepl	IsSystemTable
ExecIsUpdateTrigger	IsTable
HasAfterTrigger	IsTableFunction
HasDeleteTrigger	IsTrigger
HasInsertTrigger	IsUniqueCnst
HasInsteadOfTrigger	IsUserTable

IsView	TableHasForeignRef
OwnerId	TableHasIdentity
TableDeleteTrigger	TableHasIndex
TableDeleteTriggerCount	TableHasInsertTrigger
TableFullTextBackgroundUpdateIndexOn	TableHasNonclustIndex
TableFulltextCatalogId	TableHasPrimaryKey
TableFullTextChangeTrackingOn	TableHasRowGuidCol
TableFulltextDocsProcessed	TableHasTextImage
TableFulltextFailCount	TableHasTimestamp
TableFulltextItemCount	TableHasUniqueCnst
TableFulltextKeyColumn	TableHasUpdateTrigger
TableFulltextPendingChanges	TableInsertTrigger
TableFulltextPopulateStatus	TableInsertTriggerCount
TableHasActiveFulltextIndex	TableIsFake
TableHasCheckCnst	TableIsLockedOnBulkLoad
TableHasClustIndex	TableIsPinned
TableHasDefaultCnst	TableTextInRowLimit
TableHasDeleteTrigger	TableUpdateTrigger
TableHasForeignKey	TableUpdateTriggerCount

Vraćena vrednost iz ove funkcije će biti `1` za tačan (`True`), `0` za netačan (`False`) i `NULL` ako je ulaz bio neispravan, osim za:

- `OwnerId`: vraća ID korisnika baze podataka za vlasnika tog objekta – obratite pažnju na to da ovo nije isto što i SchemaID tog objekta i verovatno neće biti od koristi u SQL Serveru 2005 i nakon toga.
- `TableDeleteTrigger`, `TableInsertTrigger`, `TableUpdateTrigger`: vraća ID prvog okidača navedene vrste. Vraća se nula ako ne postoji okidač te vrste.
- `TableDeleteTriggerCount`, `TableInsertTriggerCount`, `TableUpdateTrigger-Count`: vraća broj okidača navedene vrste nad tabelom koja je u pitanju.
- `TableFulltextCatalogId`: vraća ID kataloga potpunog teksta ako postoji, a nulu ako za tu tabelu ne postoji katalog potpunog teksta.
- `TableFulltextKeyColumn`: vraća ColumnID kolone koja se koristi kao jedinstveni indeks za taj indeks potpunog teksta.
- `TableFulltextPendingChanges`: broj stavki koje su se promenile nakon poslednje analize potpunog teksta izvršene za ovu tabelu. Da bi ova funkcija davala valjane rezultate mora biti omogućeno praćenje promena.

- `TableFulltextPopulateStatus`: ovde ima više mogućih vraćenih vrednosti:
  - `0`: znači da proces potpunog teksta trenutno miruje.
  - `1`: trenutno je u toku potpuno popunjavanje.
  - `2`: trenutno je u toku inkrementalno popunjavanje.
  - `3`: promene se trenutno analiziraju i dodaju u katalog potpunog teksta.
  - `4`: trenutno je u toku neki vid pozadinskog ažuriranja (kao kada radi mehanizam automatskog praćenja promena).
  - `5`: u toku je operacija potpunog teksta, ali je prigušena (da bi se omogućili drugi sistemski zahtevi po potrebi) ili pauzirana.

  Povratne informacije ove opcije možete da upotrebite za donošenje odluka o tome koje druge opcije koje se odnose na potpuni tekst dolaze u obzir (ako je u toku popunjavanje znaćete da li su moguće druge funkcije, kao što je `TableFulltextDocsProcessed`).
- `TableFulltextDocsProcessed`: moguća jedino dok je aktivno indeksiranje potpunim tekstom, vraća se broj redova obrađenih od kako je započet proces indeksiranja potpunim tekstom. Rezultat nula znači da indeksiranje potpunim tekstom trenutno nije aktivno (rezultat `NULL` znači da indeksiranje potpunim tekstom nije konfigurisano za ovu tabelu).
- `TableFulltextFailCount`: moguća jedino dok je aktivno indeksiranje potpunim tekstom, vraća broj redova koji su prilikom indeksiranja potpunim tekstom, iz nekog razloga, preskočeni (nema ukazivanja na razlog). Kao za `TableFulltextDocsProcessed`: nula znači da indeksiranje potpunim tekstom trenutno nije aktivno, a `NULL` znači da indeksiranje potpunim tekstom nije konfigurisano za ovu tabelu.
- `TableIsPinned`: ovo je ostavljeno samo radi kompatibilnosti unazad, pa će u SQL Serveru 2005 i novijim uvek vratiti „0”.

OBJECTPROPERTYEX

`OBJECTPROPERTYEX` je proširena verzija funkcije `OBJECTPROPERTY`.

```
OBJECTPROPERTYEX(<id>, <property>)
```

Kao za `OBJECTPROPERTY`, parametar `id` je ID traženog objekta. Parametar `property` određuje zahtevanu informaciju o objektu. `OBJECTPROPERTYEX` podržava iste vrednosti za svojstva kao `OBJECTPROPERTY` ali su dodatno omogućene i sledeće vrednosti za `property`:

- `BaseType`: vraća osnovni tip podatka jednog objekta.
- `IsPrecise`: ukazuje na to da vaš objekat ne sadrži neprecizna izračunavanja. Na primer, `int` ili `decimal` su precizni, ali `float` nije – za izračunavanja koja koriste neprecizne tipove podatka mora se pretpostaviti da vraćaju neprecizne rezultate. Obratite pažnju na to da svaki .NET sklop koji proizvedete možete izričito da označite kao precizan.
- `IsSystemVerified`: ukazuje na to da li svojstva `IsPrecise` i `IsDeterministic` može da proveri i sam SQL Server (ili ih samo postavlja korisnik).

- `SchemaId`: kao što izgleda – vraća interni sistemski ID datog objekta. Zatim možete da upotrebite `SCHEMA_NAME` da za taj ID šeme odredite razumljivije ime.
- `SystemDataAccess`: znači da li se taj objekat oslanja na podatke neke sistemske tabele.
- `UserDataAccess`: znači da li taj objekat koristi podatke neke korisničke tabele ili podatke sistemskog korisnika.

@@PROCID

Vraća ID snimljene procedure koja se trenutno izvršava.

Pre svega alatka za ispitivanje grešaka kada se izvršava proces koji troši veliku količinu resursa. Koristi se uglavnom kao funkcija za administratore baze podataka.

SCHEMA_ID

Za dato ime šeme, vraća interni sistemski ID za tu šemu. Koristi se sintaksa:

```
SCHEMA_ID( <schema name> )
```

SCHEMA_NAME

Za dati interni sistemski ID šeme, vraća razumljivo ime za tu šemu. Sintaksa je:

```
SCHEMA_NAME( <schema id> )
```

SQL_VARIANT_PROPERTY

Funkcija `SQL_VARIANT_PROPERTY` je moćna funkcija i vraća informacije o izrazu tipa `sql_variant`. Ta informacija može da bude o svojstvima `BaseType`, `Precision`, `Scale`, `TotalBytes`, `Collation`, ili `MaxLength`. Sintaksa je:

```
SQL_VARIANT_PROPERTY (expression, property)
```

`Expression` je izraz tipa `sql_variant`. Parametar `property` može da bude bilo koja od vrednosti iz tabele B-5.

TABELA B-5: Svojstva za SQL_VARIANT PROPERTY

VREDNOST	OPIS	OSNOVNI TIP VRAĆENOG IZRAZA ZASQL_VARIANT
`BaseType`	Tipovi podataka mogu da budu `char`, `int`, `money`, `nchar`, `ntext`, `numeric`, `nvarchar`, `real`, `smalldatetime`, `smallint`, `smallmoney`, `text`, `timestamp`, `tinyint`, `uniqueidentifier`, `varbinary`, `varchar`	`Sysname`

nastavlja se

TABELA B-5 *(nastavak)*

VREDNOST	OPIS	OSNOVNI TIP VRAĆENOG IZRAZA ZASQL_VARIANT
`Precision`	Preciznost numeričkog osnovnog tipa podatka: `datetime` = 23 `smalldatetime` = 16 `float` = 53 `real` = 24 `decimal` (p,s) i `numeric` (p,s) = p `money` = 19 `smallmoney` = 10 `int` = 10 `smallint` = 5 `tinyint` = 3 `bit` = 1 Svi ostali tipovi = 0	`Int`
`Scale`	Broj cifara desno od decimalnog zareza numeričkih osnovnih tipova podataka: `decimal` (p,s) i `numeric` (p,s) = s `money` i `smallmoney` = 4 `datetime` = 3 svi ostali tipovi = 0	`Int`
`TotalBytes`	Broj bajtova potrebnih da se prime metapodaci i podaci vrednosti. Ako je to veće od 900, indeks neće moći da se pravi.	`Int`
`Collation`	Redosled sravnjivanja konkretne vrednosti `sql_variant`.	`Sysname`
`MaxLength`	Maksimalna dužina tipa podatka, u bajtovima.	`int`

TYPEPROPERTY

Funkcija `TYPEPROPERTY` vraća informacije o tipovima podataka. Sintaksa je sledeća:

```
TYPEPROPERTY(<type>, <property>)
```

Parametar `type` je ime tipa podatka. Parametar `property` je svojstvo tipa podatka koje se ispituje; može da bude neka od sledećih vrednosti:

- `Precision`: vraća broj cifara/znakova.
- `Scale`: vraća broj decimalnih mesta.

- `AllowsNull`: vraća `1` za tačan (`True`) ako dozvoljava null, a `0` za netačan (`False`) ako ne dozvoljava.
- `UsesAnsiTrim`: vraća `1` za tačan (`True`), a `0` za netačan (`False`) ako ne koristi `AnsiTrim`.

FUNKCIJE SKUPA REDOVA

Funkcije skupa redova (rowset) vraćaju objekat koji može da se koristi umesto reference tabele u T-SQL naredbama. Funkcije skupa redova su:

- `CHANGETABLE`
- `CONTAINSTABLE`
- `FREETEXTTABLE`
- `OPENDATASOURCE`
- `OPENQUERY`
- `OPENROWSET`
- `OPENXML`

CHANGETABLE

Vraća informacije o praćenju promena za tabelu. Ovom naredbom možete da dobijete sve promene za tabelu, ili informacije o praćenju promena za konkretan red. Sintaksa je sledeća:

```
CHANGETABLE (
        { CHANGES table , last_sync_version
        | VERSION table , <primary_key_values> } )
[AS] table_alias [ ( column_alias [ ,...n ] )
```

CONTAINSTABLE

Funkcija `CONTAINSTABLE` se koristi u upitima potpunog teksta. Sintaksa je sledeća:

```
CONTAINSTABLE (<table>, {<column> | *}, '<contains_search_condition>')
```

FREETEXTTABLE

Funkcija `FREETEXTTABLE` se koristi u upitima potpunog teksta. Sintaksa je sledeća:

```
FREETEXTTABLE (<table>, {<column> | *}, '<freetext_string>')
```

OPENDATASOURCE

Funkcija `OPENDATASOURCE` obezbeđuje informacije o ad hok konekciji. Sintaksa je sledeća:

```
OPENDATASOURCE (<provider_name>, <init_string>)
```

provider_name je ime registrovano kao ProgID u OLE DB dobavljaču koji se koristi za pristupanje izvoru podataka. Parametar init_string bi trebalo da je poznat VB programerima, pošto je to inicijalizacioni string za OLE DB dobavljača. Na primer, init_string bi mogao da izgleda ovako:

```
„User Id=wonderison;Password=JuniorBlues;DataSource=MyServerName"
```

OPENQUERY

Funkcija OPENQUERY izvršava konkretan prolazni upit (pass-through query) nad navedenim serverom linked_server. Sintaksa je sledeća:

```
OPENQUERY(<linked_server>, '<query>')
```

OPENROWSET

Funkcija OPENROWSET pristupa udaljenim podacima iz OLE DB izvora podataka. Sintaksa je sledeća:

```
OPENROWSET('<provider_name>'
   {
    '<datasource>';'<user id>';'<password>'
    | '<provider_string>'
   },
   {
      [<catalog.>][<schema.>]<object>
      | '<query>'
   })
```

Parametar provider_name je string koji predstavlja razumljivo ime za OLE DB kako je zavedeno u bazi Registry. Parametar data_source je string koji odgovara zahtevanom OLE DB izvoru podataka. Parametar user_id je relevantno ime korisnika koje se prosleđuje OLE DB dobavljaču. Parametar password je lozinka pridružena uz user_id.

Parametar provider_string je string za konekciju koji je specifičan za dobavljača, a koristi se umesto kombinacije datasource, user_id, password.

Parametar catalog je ime kataloga/baze podataka koja sadrži zahtevani objekat. Parametar schema je ime vlasnika šeme ili objekta za zahtevani objekat. Parametar object je ime objekta.

Parametar query je string koji će dobavljač izvršiti, a koristi se umesto kombinacije catalog, schema, object.

OPENXML

Ako se XML dokument preda kao parametar, ili ako se izdvoji XML dokument i definiše dokument unutar promenljive, OPENXML omogućava da se ispita struktura i vrate podaci, kao da je taj XML dokument tabela. Sintaksa je sledeća:

```
OPENXML(<idoc int> [in],<rowpattern> nvarchar[in],[<flags> byte[in]])
[WITH (<SchemaDeclaration> | <TableName>)]
```

Parametar `idoc_int` je promenljiva definisana pomoću sistemske snimljene procedure `sp_xml_prepareddocument`. `Rowpattern` je definicija čvora. Parametar `flags` određuje preslikavanje između XML dokumenta i skupa redova koji treba da se vrati u naredbu `SELECT`. `SchemaDeclaration` definiše XML šemu za XML dokument; ako je u bazi podataka definisana tabela u skladu sa XML šemom, umesto šeme može da se upotrebi `TableName`.

Da bi XML dokument mogao da se koristi, morate najpre da ga pripremite pomoću sistemske procedure `sp_xml_preparedocument`.

BEZBEDNOSNE FUNKCIJE

Bezbednosne funkcije vraćaju informacije o korisnicima i ulogama. To su:

- `HAS_DBACCESS`
- `IS_MEMBER`
- `IS_SRVROLEMEMBER`
- `SUSER_ID`
- `SUSER_NAME`
- `SUSER_SID`
- `USER`
- `USER_ID`
- `USER_NAME`

HAS_DBACCESS

Funkcija `HAS_DBACCESS` utvrđuje da li korisnik koji je prijavljen ima pristup bazi podataka koja se koristi. Vraćena vrednost `1` znači da korisnik ima pristup, a vraćena vrednost `0` znači da nema. Ako se vrati `NULL` to znači da podneto ime `database_name` nije ispravno. Sintaksa je sledeća:

```
HAS_DBACCESS ('<database_name>')
```

IS_MEMBER

Funkcija `IS_MEMBER` vraća informaciju o tome da li je tekući korisnik član navedene Windows NT grupe, odnosno SQL Server uloge. Sintaksa je sledeća:

```
IS_MEMBER ({'<group>' | '<role>'})
```

Parametar `group` je ime NT grupe i mora da bude oblika `domen\grupa`. Parametar `role` je ime SQL Server uloge. Uloga može da bude fiksna uloga baze podataka ili korisnički definisana uloga ali ne može da bude serverska uloga.

Funkcija vraća vrednost `1` ako je tekući korisnik član navedene grupe ili uloge, `0` ako tekući korisnik nije član navedene grupe ili uloge, a `NULL` ako je navedeno ime grupe ili uloge neispravno.

IS_SRVROLEMEMBER

Funkcija `IS_SRVROLEMEMBER` vraća informaciju o tome da li je korisnik član navedene serverske uloge. Sintaksa je sledeća:

```
IS_SRVROLEMEMBER ('<role>' [,'<login>'])
```

Neobavezan parametar `login` je ime naloga za prijavljivanje za koji se traži provera – podrazumeva se tekući korisnik. Parametar `role` je serverska uloga i mora da ima jednu od sledećih mogućih vrednosti:

- `sysadmin`
- `dbcreator`
- `diskadmin`
- `processadmin`
- `serveradmin`
- `setupadmin`
- `securityadmin`

Ova funkcija vraća `1` ako je navedeni nalog član navedene uloge, `0` ako nalog nije član te uloge, a `NULL` ako je uloga ili ime naloga neispravno.

SUSER_ID

Funkcija `SUSER_ID` vraća ID broj logina za navedenog korisnika. Sintaksa je sledeća:

```
SUSER_ID(['<login>'])
```

Parametar `login` je ime navedenog korisnika. Ako nije data vrednost za `login`, umesto toga će se upotrebiti podrazumevana vrednost tekućeg korisnika.

NAPOMENA *Sistemsku funkciju* `SUSER_ID` *odavno je zamenila* `SUSER_SID`*, a zadržava se u proizvodu čisto u svrhu kompatibilnosti unazad. Izbegavajte* `SUSER_ID`*, pošto vraćena interna vrednost može da se menja od servera do servera (SID je mnogo pouzdaniji kad se uzme u obzir da baza podataka može da se obnovi na drugom serveru gde za dati login može da se dobije drugi* `SUSER_ID`*).*

SUSER_NAME

Funkcija `SUSER_NAME` vraća ime za navedeni ID korisničkog prijavljivanja. Sintaksa je sledeća:

```
SUSER_NAME([<server user id>])
```

Parametar `server_user_id` je ID broj prijavljivanja navedenog korisnika. Ako se ne podnese `server_user_id`, umesto toga će se upotrebiti podrazumevana vrednost tekućeg korisnika.

NAPOMENA *Sistemska funkcija* `SUSER_NAME` *uključena je u SQL Server 2012 samo u svrhu kompatibilnosti unazad, pa ako je moguće, trebalo bi umesto nje da koristite* `SUSER_SNAME`.

SUSER_SID

Funkcija `SUSER_SID` vraća bezbednosni identifikacioni broj (SID) za navedenog korisnika. Sintaksa je sledeća:

```
SUSER_SID(['<login>'])
```

Parametar `login` je korisničko ime za prijavljivanje. Ako se ne podnese vrednost za login, umesto toga će se upotrebiti tekući korisnik.

SUSER_SNAME

Funkcija `SUSER_SNAME` vraća login ID ime za navedeni bezbednosni identifikacioni broj (SID). Sintaksa je sledeća:

```
SUSER_SNAME([<server user sid>])
```

Parametar `server_user_sid` je korisnikov SID. Ako se ne podnese vrednost za `server_user_sid`, umesto toga će se upotrebiti vrednost tekućeg korisnika.

USER

Funkcija `USER` omogućava da se u tabelu insertuje sistemski dodeljena vrednost za korisničko ime tekućeg korisnika baze podataka, ako nije predviđena podrazumevana vrednost. Sintaksa je sledeća:

```
USER
```

USER_ID

Funkcija `USER_ID` vraća ID broj u bazi podataka za navedenog korisnika. Sintaksa je sledeća:

```
USER_ID(['<user>'])
```

Parametar `user` je ime korisnika koje treba da se upotrebi. Ako se ne podnese vrednost za `user`, upotrebiće se tekući korisnik.

USER_NAME

Funkcija `USER_NAME` je funkcionalno obrnuta funkciji `USER_ID`, pa vraća korisničko ime navedenog korisnika u bazi podataka za dati ID broj u bazi podataka. Sintaksa je sledeća:

```
USER_NAME(['<user id>'])
```

Parametar `user id` je ID korisnika čije ime tražite. Ako se ne podnese vrednost za `user id`, podrazumeva se tekući korisnik.

FUNKCIJE ZA STRING

Funkcije za string vrše akcije nad string vrednostima i vraćaju stringove ili numeričke vrednosti. Funkcije za string su:

- `ASCII`
- `CHAR`
- `CHARINDEX`
- `CONCAT`
- `DIFFERENCE`
- `FORMAT`
- `LEFT`
- `LEN`
- `LOWER`
- `LTRIM`
- `NCHAR`
- `PATINDEX`
- `QUOTENAME`
- `REPLACE`
- `REPLICATE`
- `REVERSE`
- `RIGHT`
- `RTRIM`
- `SOUNDEX`
- `SPACE`

- STR
- STUFF
- SUBSTRING
- UNICODE
- UPPER

ASCII

Funkcija ASCII vraća vrednost u ASCII kodu za prvi znak na levom kraju izraza character_expression. Sintaksa je sledeća:

```
ASCII(<character expression>)
```

CHAR

Funkcija CHAR konvertuje ASCII kôd (naveden u expression) u string. Sintaksa je sledeća:

```
CHAR(<expression>)
```

Izraz expression može da bude bilo koji ceo broj od 0 do 255.

CHARINDEX

Funkcija CHARINDEX vraća početnu poziciju izraza expression unutar stringa character_string. Sintaksa je sledeća:

```
CHARINDEX(<expression>, <character string> [, <start location>])
```

Parametar expression je string koji se traži. character_string je string koji se pretražuje, obično kolona. Parametar start_location je pozicija od koje se traženje počinje, ako to nije pozitivan broj, traženje će početi na početku stringa character_string.

CONCAT

Nova u SQL Serveru 2012, funkcija CONCAT predstavlja nov način sklapanja stringa od delova, jednostavniji za upotrebu od operatora '+'. Sa CONCAT, vi podnesete dva ili više parametra – to mogu da budu stringovi, ili bilo šta što može implicitno da se konvertuje u string – a CONCAT će ih spojiti u string, vršeći pre toga sve potrebne konverzije tipova podataka.

```
CONCAT ( string_value1, string_value2 [, string_valueN ] )
```

DIFFERENCE

Funkcija DIFFERENCE vraća, kao ceo broj, razliku između SOUNDEX vrednosti dva izraza. Sintaksa je sledeća:

```
DIFFERENCE(<expression1>, <expression2>)
```

Ova funkcija vraća ceo broj između `0` i `4`. Ako dva izraza zvuče identično (na primer, blue i blew) vratiće se vrednost `4`. Ako nema sličnosti, vratiće se vrednost `0`.

FORMAT

`FORMAT` prihvata podatke u većini numeričkih ili datumskih/vremenskih tipova podataka i vraća formatiran string prema zadatom parametru formata. Parametar formata koristi iste stringove formata koji se koriste za .NET, tako da su veoma raznovrsni, a mogućnosti ima toliko da neću ovde da ih navodim. Potražite „Formatting Types" u Books Online ili u MSDN-u za kompletnu onlajn referencu.

```
FORMAT ( value, format [, culture ] )
```

LEFT

Funkcija `LEFT` vraća krajnje levi deo izraza, navedeni broj znakova od levog kraja. Sintaksa je sledeća:

```
LEFT(<expression>, <integer>)
```

Parametar `expression` sadrži znakovne podatke iz kojih će se izdvojiti krajnje levi deo. Parametar `integer` određuje broj znakova od levog kraja koji se uzimaju – mora da bude pozitivan ceo broj.

LEN

Funkcija `LEN` vraća broj znakova u navedenom izrazu `expression`. Sintaksa je sledeća:

```
LEN(<expression>)
```

LOWER

Funkcija `LOWER` konvertuje sva velika slova u izrazu `expression` u mala slova. Sintaksa je sledeća:

```
LOWER(<expression>)
```

LTRIM

Funkcija `LTRIM` uklanja vodeće nule iz znakovnog izraza `character_expression`. Sintaksa je sledeća:

```
LTRIM(<character_expression>)
```

NCHAR

Funkcija `NCHAR` vraća Unicode znak navedene vrednosti `integer_code`. Sintaksa je sledeća:

```
NCHAR(<integer_code>)
```

Parametar `integer_code` mora da bude pozitivan ceo broj od 0 do 65.535.

PATINDEX

Funkcija `PATINDEX` vraća početnu poziciju prvog pojavljivanja jednog šablona u navedenom izrazu, ili nulu ako se šablon ne pronađe. Sintaksa je sledeća:

```
PATINDEX('<%pattern%>', <expression>)
```

Parametar `pattern` je string koji se traži. Dozvoljeni su džokerski znaci, ali oko šablona moraju da stoje znaci `%`. Parametar `expression` su znakovni podaci u kojima se šablon traži – obično kolona.

QUOTENAME

Funkcija `QUOTENAME` vraća Unicode string sa dodatim graničnicima kako bi navedeni string bio važeći ograničeni identifikator za SQL Server. Sintaksa je sledeća:

```
QUOTENAME('<character_string>'[, '<quote_character>'])
```

Parametar `character_string` je Unicode string. Parametar `quote_character` je string od jednog znaka koji će se koristiti kao graničnik. Parametar `quote_character` može biti jednostruki navodnik (`'`), leva ili desna uglasta zagrada (`[]`), ili dvostruki navodnik (`"`) – podrazumevaju se uglaste zagrade.

REPLACE

Funkcija `REPLACE` sva pojavljivanja drugog navedenog stringa u prvom navedenom stringu zamenjuje trećim navedenim stringom. Sintaksa je sledeća:

```
REPLACE('<string_expression1>', '<string_expression2>',
'<string_expression3>')
```

Parametar `string_expression1` je izraz u kojem se vrši traženje. Parametar `string_expression2` je izraz koji se traži u izrazu `string_expression1`. Parametar `string_expression3` je izraz kojim se zamenjuje svako pojavljivanje izraza `string_expression2`.

REPLICATE

Funkcija `REPLICATE` ponavlja izraz `character_expression` navedeni broj puta. Sintaksa je sledeća:

```
REPLICATE(<character_expression>, <integer>)
```

REVERSE

Funkcija `REVERSE` vraća obrnut izraz `character_expression`. Sintaksa je sledeća:

```
REVERSE(<character_expression>)
```

RIGHT

Funkcija `RIGHT` vraća krajnje desni deo, navedeni broj znakova (naveden sa `integer`) od desnog kraja izraza `character_expression`. Sintaksa je sledeća:

```
RIGHT(<character_expression>, <integer>)
```

Parametar `integer` mora da bude pozitivan ceo broj.

RTRIM

Funkcija `RTRIM` uklanja blenkove na kraju `character_expression`. Sintaksa je sledeća:

```
RTRIM(<character_expression>)
```

SOUNDEX

Funkcija `SOUNDEX` vraća četvoroznačni (SOUNDEX) kôd, koji može da se upotrebi za procenjivanje sličnosti dva stringa. Sintaksa je sledeća:

```
SOUNDEX(<character_expression>)
```

SPACE

Funkcija `SPACE` vraća string ponovljenih razmaka, čija dužina se zadaje parametrom `integer`. Sintaksa je sledeća:

```
SPACE(<integer>)
```

STR

Funkcija `STR` konvertuje numeričke podatke u znakovne podatke. Sintaksa je sledeća:

```
STR(<numeric_expression>[, <length>[, <decimal>]])
```

Parametar `numeric_expression` je numerički izraz sa decimalnom tačkom. Parametar `length` je ukupna dužina uključujući decimalnu tačku, cifre i razmake. Parametar `decimal` je broj mesta desno od decimalne tačke.

STUFF

Funkcija `STUFF` briše znakove u navedenoj dužini i insertuje drugi skup znakova na njihovo mesto. Sintaksa je sledeća:

```
STUFF(<expression>, <start>, <length>, <characters>)
```

Parametar `expression` je string znakova od kojih će se neki brisati a drugi dodavati. Parametar `start` određuje gde se počinje brisanje i insertovanje znakova. Parametar `length` određuje broj znakova koji se brišu. Parametar `characters` je novi skup znakova koji se insertuju u `expression`.

SUBSTRING

Funkcija `SUBSTRING` vraća deo izraza. Sintaksa je sledeća:

```
SUBSTRING(<expression>, <start>, <length>)
```

Parametar `expression` određuje podatke iz kojih će se uzeti podstring. To može da bude znakovni string, binarni string, tekst, ili izraz koji sadrži tabelu. Parametar `start` je ceo broj koji određuje gde počinje postring. Parametar `length` određuje dužinu podstringa.

UNICODE

Funkcija `UNICODE` vraća Unicode broj koji predstavlja prvi znak u `character_expression`. Sintaksa je sledeća:

```
UNICODE('<character_expression>')
```

UPPER

Funkcija `UPPER` sva mala slova u `character_expression` konvertuje u velika slova. Sintaksa je sledeća:

```
UPPER(<character_expression>)
```

SISTEMSKE FUNKCIJE

Sistemske funkcije – stariji naziv za ono što Microsoft sada jednostavno zove „ostale" – mogu da se koriste za vraćanje informacija o vrednostima, objektima i podešavanjima u SQL Serveru. Funkcije su sledeće:

- `APP_NAME`
- `CASE`
- `CAST` and `CONVERT`

- COALESCE
- COLLATIONPROPERTY
- CURRENT_TIMESTAMP
- CURRENT_USER
- DATALENGTH
- FORMATMESSAGE
- GETANSINULL
- HOST_ID
- HOST_NAME
- IDENT_CURRENT
- IDENT_INCR
- IDENT_SEED
- IDENTITY
- ISDATE
- ISNULL
- ISNUMERIC
- NEWID
- NULLIF
- PARSENAME
- PERMISSIONS
- ROWCOUNT_BIG
- SCOPE_IDENTITY
- SERVERPROPERTY
- SESSION_USER
- SESSIONPROPERTY
- STATS_DATE
- SYSTEM_USER

APP_NAME

Funkcija `APP_NAME` vraća ime aplikacije za tekuću sesiju, ako je bilo postavljeno u aplikaciji kao tip `nvarchar`. Sintaksa je sledeća:

```
APP_NAME()
```

CASE

Funkcija CASE procenjuje listu uslova i vraća jedan od više mogućih rezultata. Osim toga, ona ima dva formata:

- Jednostavna funkcija CASE poredi jedan izraz sa skupom jednostavnih izraza da bi odredila rezultat.
- Tražena funkcija CASE procenjuje skup Bulovskih izraza da bi odredila rezultat.

NAPOMENA *Oba formata podržavaju neobavezan argument* ELSE.

Jednostavna funkcija CASE:

```
CASE <input expression>
   WHEN <when expression> THEN <result expression>
   ELSE <else result expression>
END
```

Tražena funkcija CASE:

```
CASE
    WHEN <Boolean expression> THEN <result expression>
    ELSE <else result expression>
END
```

COALESCE

Funkciji COALESCE se predaje neodređen broj argumenata, a ona među njima traži prvi izraz koji nije null. Sintaksa je sledeća:

```
COALESCE(<expression> [,...n])
```

Ako su svi argumenti NULL tada COALESCE vraća NULL.

COLLATIONPROPERTY

Funkcija COLLATIONPROPERTY vraća svojstvo datog sortiranja reči (collation). Sintaksa je sledeća:

```
COLLATIONPROPERTY(<collation_name>, <property>)
```

Parametar collation_name je ime sortiranja reči koje hoćete da koristite, a property je svojstvo sortiranja koje hoćete da utvrdite. To može biti jedna od sledeće tri vrednosti:

- **CodePage**: Kodna stranica sortiranja koja nije Unicode.

- **LCID**: Windows LCID za to sortiranje. Vraća NULL za SQL sortiranja.
- **ComparisonStyle**: Windows stil sravnjivanja tog sortiranja. Vraća NULL za binarna ili SQL sortiranja.

CURRENT_USER

Funkcija CURRENT_USER jednostavno vraća tekućeg korisnika kao tip sysname. To je ekvivalentno sa USER_NAME(). Sintaksa je sledeća:

```
CURRENT_USER
```

DATALENGTH

Funkcija DATALENGTH vraća broj bajtova potrebnih da se expression predstavi kao ceo broj. To je posebno korisno za tipove podataka varchar, varbinary, text, image, nvarchar i ntext zato što se u tim tipovima podataka mogu čuvati podaci varijabilne dužine. Sintaksa je sledeća:

```
DATALENGTH(<expression>)
```

@@ERROR

Vraća šifru greške poslednje T-SQL naredbe izvršene na tekućoj konekciji. Ako nije bilo greške, vrednost će biti nula.

Ako nameravate da pišete snimljene procedure ili okidače, ovo je jedna od najbitnijih sistemskih funkcija – takoreći nećete preživeti bez nje.

NAPOMENA *Ne smete da zaboravite da je životni vek za* @@ERROR *samo jedna naredba. To znači da, ako pomoću nje hoćete da proverite da li je došlo do greške u nekoj naredbi, morate to da ispitate odmah u sledećoj naredbi, ili da premestite vrednost* @@ERROR *u neku pomoćnu promenljivu. Uglavnom, ja preporučujem da se koristi* ERROR_NUMBER() *u bloku* TRY...CATCH *osim ako morate da podržite neki kôd stariji od SQL Servera 2005.*

Lista svih sistemskih grešaka može se videti u sistemskoj tabeli sys.messages u bazi podataka master.

Ako hoćete da pravite vlastite prilagođene greške, upotrebite sp_addmessage.

FORMATMESSAGE

Funkcija FORMATMESSAGE konstruiše poruku koristeći postojeće poruke u sysmessages. Sintaksa je sledeća:

```
FORMATMESSAGE(<msg_number>, <param value>[,...n])
```

Gde je msg_number ID poruke u sysmessages.

NAPOMENA `FORMATMESSAGE` *traži poruku u tekućem jeziku korisnika. Ako ne postoji lokalizovana verzija poruke, upotrebiće se verzija U.S. English.*

GETANSINULL

Funkcija `GETANSINULL` vraća, kao ceo broj, podrazumevanu mogućnost vrednosti `NULL` (nullability) za bazu podataka. Sintaksa je sledeća:

```
GETANSINULL(['<database>'])
```

Parametar `database` je ime baze podataka za koju treba vratiti informaciju o mogućnosti vrednosti `NULL`.

Kada su za datu bazu podataka dozvoljene `NULL` vrednosti, a ta mogućnost za kolonu ili tip podatka nije eksplicitno definisana, `GETANSINULL` vraća `1`. To je podrazumevna vrednost za `ANSI NULL`.

HOST_ID

Funkcija `HOST_ID` vraća ID radne stanice. Sintaksa je sledeća:

```
HOST_ID()
```

HOST_NAME

Funkcija `HOST_NAME` vraća ime radne stanice. Sintaksa je sledeća:

```
HOST_NAME()
```

IDENT_CURRENT

Funkcija `IDENT_CURRENT` vraća poslednju identitetsku vrednost napravljenu za tabelu, u bilo kojoj sesiji ili opsegu te tabele. To je isto kao `@@IDENTITY` i `SCOPE_IDENTITY`; međutim, ovde nema ograničenja dometa traženja vrednosti koja treba da se vrati.

Sintaksa je sledeća:

```
IDENT_CURRENT('<table_name>')
```

Parametar `table_name` je tabela za koju hoćete da nađete tekući identitet.

IDENT_INCR

Funkcija `IDENT_INCR` vraća vrednost inkrementa određenog prilikom pravljenja identitetske kolone u tabeli ili prikazu koji ima identitetsku kolonu. Sintaksa je sledeća:

```
IDENT_INCR('<table_or_view>')
```

Parametar `table_or_view` je izraz kojim se određuje tabela ili prikaz u kojem se traži valjana vrednost inkrementa za identitet.

IDENT_SEED

Funkcija `IDENT_SEED` vraća vrednost semena određenu prilikom pravljenja identitetske kolone u tabeli ili prikazu koji ima identitetsku kolonu. Sintaksa je sledeća:

```
IDENT_SEED('<table_or_view>')
```

Parametar `table_or_view` je izraz kojim se određuje tabela ili prikaz u kojem se traži valjana vrednost semena.

@@IDENTITY

Vraća poslednju identitetsku vrednost napravljenu u tekućoj konekciji.

Ako koristite identitetske kolone a zatim ih u drugoj tabeli referencirate kao strani ključ, stalno ćete koristiti ovu funkciju. Možete da napravite nadređeni zapis (obično onaj sa identitetom koji treba da izdvojite), a zatim selektirate `@@IDENTITY` da biste znali za koju vrednost treba da vežete podređene zapise.

Ako vršite insertovanja u više tabela sa identitetskim vrednostima, ne zaboravite da se u `@@IDENTITY` nalazi samo *poslednja* insertovana identitetska vrednost – sve pre toga je izgubljeno, ukoliko vrednost ne prenesete u pomoćnu promenljivu nakon svakog inserta. Osim toga, ako poslednja tabela u koju ste insertovali nema identitetsku kolonu, `@@IDENTITY` će imati vrednost `NULL`.

IDENTITY

Funkcija `IDENTITY` se koristi za insertovanje identitetske kolone u novu tabelu. Koristi se jedino sa `SELECT` naredbom sa klauzulom `INTO` za tabelu. Sintaksa je sledeća:

```
IDENTITY(<data type>[, <seed>, <increment>]) AS <column name>
```

Gde je:

- `data type` tip podatka identitetske kolone.
- `seed` vrednost koju treba dodeliti prvom redu u tabeli. Svakom sledećem redu se dodeljuje sledeća identitetska vrednost, koja je jednaka poslednjoj vrednosti `IDENTITY` uvećanoj za vrednost `increment`. Ako se ne navedu ni `seed` niti `increment`, za oboje se podrazumeva vrednost `1`.
- `increment` inkrement koji se dodaje na vrednost `seed` za uzastopne redove u tabeli.
- `column name` ime kolone koja treba da se insertuje u novu tabelu.

ISNULL

Funkcija `ISNULL` ispituje da li izraz ima vrednost `NULL` i u tom slučaju je zamenjuje navedenom vrednošću za zamenu. Sintaksa je sledeća:

```
ISNULL(<check expression>, <replacement value>)
```

ISNUMERIC

Funkcija `ISNUMERIC` utvrđuje da li je izraz važećeg numeričkog tipa. Sintaksa je sledeća:

```
ISNUMERIC(<expression>)
```

NEWID

Funkcija `NEWID` pravi jedinstvenu vrednost tipa `uniqueidentifier`. Sintaksa je sledeća:

```
NEWID()
```

NULLIF

Funkcija `NULLIF` poredi dva izraza i vraća `NULL` vrednost ako su izrazi jednaki. Sintaksa je sledeća:

```
NULLIF(<expression1>, <expression2>)
```

PARSENAME

Funkcija `PARSENAME` vraća navedeni deo imena objekta. Sintaksa je sledeća:

```
PARSENAME('<object_name>', <object_piece>)
```

Parametar `object_name` je ime objekta iz kojeg treba izvaditi deo imena. Parametar `object_piece` određuje koji deo imena objekta treba vratiti. Parametar `object_piece` može imati jednu od sledećih vrednosti:

- `1`: ime objekta
- `2`: ime vlasnika
- `3`: ime baze podataka
- `4`: ime servera

PERMISSIONS

Funkcija `PERMISSIONS` vraća vrednost koja sadrži bitmapu, koja za tekućeg korisnika pokazuje dozvole za naredbu, objekat, ili kolonu. Sintaksa je sledeća:

```
PERMISSIONS([<object_id> [, '<column>']])
```

Parametar `object_id` je ID objekta. Neobavezan parametar `column` je ime kolone za koju se vraća informacija o dozvolama.

@@ROWCOUNT

Vraća broj redova pogođenih poslednjom naredbom.

Globalna promenljiva koja se najviše koristi, ja je najčešće koristim za proveravanje grešaka koje ne prekidaju izvršavanje – to jest, za logičke greške u programu koje za SQL Server ne predstavljaju nikakav problem. Kada, na primer, imate situaciju sa ažuriranjem koje zavisi od uslova, ali nalazite da je pogođeno nula redova. Ako je vaš klijent podneo izmenu za konkretan red, on verovatno očekuje da taj red zadovoljava date kriterijume – nula pogođenih redova znači da nešto nije u redu.

Međutim, ako ispitate ovu sistemsku funkciju za naredbu koja ne vraća redove, takođe će se vratiti vrednost `0`.

ROWCOUNT_BIG

Funkcija `ROWCOUNT_BIG` je veoma slična funkciji `@@ROWCOUNT` po tome što vraća broj redova iz poslednje naredbe. Međutim, vraćena vrednost je tipa `bigint`. Sintaksa je sledeća:

```
ROWCOUNT_BIG()
```

SCOPE_IDENTITY

Funkcija `SCOPE_IDENTITY` vraća poslednju vrednost insertovanu u identitetsku kolonu u istom opsegu (to jest, unutar iste snimljene procedure, okidača, funkcije, ili paketa). To je slično funkciji `IDENT_CURRENT`, koju smo već opisali, mada ona nije ograničena na insertovanja identiteta u istom opsegu.

Ova funkcija vraća tip podatka `sql_variant`, a sintaksa je sledeća:

```
SCOPE_IDENTITY()
```

SERVERPROPERTY

Funkcija `SERVERPROPERTY` vraća informacije o serveru na kojem radite. Sintaksa je sledeća:

```
SERVERPROPERTY('<propertyname>')
```

Moguće vrednosti za `propertyname` nalaze se u tabeli B-6.

TABELA B-6: Vrednosti propertyname

IME SVOJSTVA	VRAĆENE VREDNOSTI
`Collation`	Ime podrazumevanog sortnog redosleda na serveru.
`Edition`	Izdanje primerka SQL Servera instaliranog na serveru. Vraća jedan od sledećih `nvarchar` rezultata: `'Desktop Engine''Developer Edition''Enterprise Edition''Enterprise Evaluation Edition''Personal Edition''Standard Edition'`.
`Engine Edition`	Izdanje mašine primerka SQL Servera instaliranog na serveru: `1` – Personal ili Desktop Engine, `2` – Standard, ili `3` – Enterprise (vraća se za izdanja Enterprise, Enterprise Evaluation i za Developer).

IME SVOJSTVA	VRAĆENE VREDNOSTI
`InstanceName`	Ime primerka sa kojim je korisnik konektovan.
`IsClustered`	Utvrđuje da li je primerak servera konfigurisan u klaster za premošćavanje otkaza: `1` – Klasterovan, `0` – Nije klasterovan, a `NULL` – neispravan ulaz.
`IsFullTextInstalled`	Da se utvrdi da li je sa tekućim primerkom SQL Servera instalirana komponenta potpunog teksta: `1` – Potpuni tekst je instaliran, `0` – Potpuni tekst nije instaliran, a `NULL` – neispravan ulaz.
`IsIntegratedSecurityOnly`	Da se utvrdi da li je server u integrisanom bezbednosnom režimu: `1` – Integrisana bezbednost, `0` – Nije integrisana bezbednost, a `NULL` – neispravan ulaz.
`IsSingleUser`	Da se utvrdi da li je server instalacija za jednog korisnika: `1` – Jedan korisnik, `0` – Nije jedan korisnik, a `NULL` – neispravan ulaz.
`IsSyncWithBackup`	Da se utvrdi da li je baza podataka objavljena ili distribuciona baza podataka koja može da se obnavlja bez remećenja trenutne transakcione replikacije: `1` – tačno (`True`) ili `0` – netačno (`False`).
`LicenseType`	Koja vrsta licence je instalirana za ovaj primerak SQL Servera: `PER_SEAT` – režim po korisniku, `PER_PROCESSOR` – režim po procesoru, `DISABLED` – licenciranje je onemogućeno.
`MachineName`	Vraća Windows NT ime računara na kojem se izvršava primerak servera. Za klasterovan primerak (primerak SQL Servera koji se izvršava na virtuelnom serveru na Microsoft Cluster Serveru), vraća ime virtuelnog servera.
`NumLicenses`	Broj klijentskih licenci registrovanih za ovaj primerak SQL Servera, ako je režim po korisniku. Broj procesora licenciranih za ovaj primerak SQL Servera, ako je režim po procesoru.
`ProcessID`	ID procesa SQL Server servisa. (`ProcessID` je korisno za prepoznavanje koji `sqlservr.exe` pripada ovom primerku.)
`ProductVersion`	Veoma slično Visual Basic projektima, po tome što se vraćaju detalji verzije primerka SQL Servera, u obliku `'major.minor.build'`.
`ProductLevel`	Vraća vrednost verzije primerka SQL Servera koji se trenutno izvršava. Vraća: `'RTM'` – verzija Shipping `'SPn'` – verzija Service pack `'Bn'` – verzija Beta
`ServerName`	Informacije o Windows NT serveru i o primerku za navedeni primerak SQL Servera.

NAPOMENA *Funkcija SERVERPROPERTY je veoma korisna za korporacije sa više sajtova gde su programerima potrebne informacije sa servera.*

SESSION_USER

Funkcija `SESSION_USER` omogućava da se u tabelu insertuje sistemski ponuđena vrednost za korisničko ime tekuće sesije ako nije bila data podrazumevana vrednost. Sintaksa je sledeća:

```
SESSION_USER
```

SESSIONPROPERTY

Funkcija `SESSIONPROPERTY` se koristi za vraćanje `SET` opcija za sesiju. Sintaksa je sledeća:

```
SESSIONPROPERTY (<option>)
```

Ova funkcija je korisna kada snimljene procedure menjaju svojstva sesije u specifičnim scenarijima. Ovu funkciju ne bi trebalo često koristiti, pošto ne biste smeli da menjate previše `SET` opcija za vreme izvršavanja.

STATS_DATE

Funkcija `STATS_DATE` vraća datum kada su poslednji put ažurirane statistike za navedeni indeks. Sintaksa je sledeća:

```
STATS_DATE(<table id>, <index id>)
```

SYSTEM_USER

Funkcija `SYSTEM_USER` omogućava da se u tabelu insertuje sistemski ponuđena vrednost za tekuće sistemsko korisničko ime ako nije bila data podrazumevana vrednost. Sintaksa je sledeća:

```
SYSTEM_USER
```

@@TRANCOUNT

Vraća broj aktivnih transakcija – u suštini, nivo ugnežđavanja transakcija – za tekuću konekciju.

Ova funkcija je jako važna kada koristite transakcije. Ja nisam ljubitelj ugnežđenih transakcija, ali ponekad ih je teško izbeći. Zato može da bude važno saznati gde se nalazite u pogledu ugnežđavanja transakcija (na primer, možete imati logiku da se započne transakcija, samo ako već niste u transakciji).

Ako niste u transakciji, `@@TRANCOUNT` je jednako `0`. Pogledajte jedan kratak primer:

```
SELECT @@TRANCOUNT As TransactionNestLevel     --To će biti nula
                                               --na ovom mestu

BEGIN TRAN
SELECT @@TRANCOUNT As TransactionNestLevel     --To će biti jedan
                                               --na ovom mestu

  BEGIN TRAN
    SELECT @@TRANCOUNT As TransactionNestLevel --To će biti dva
                                               --na ovom mestu

  COMMIT TRAN
SELECT @@TRANCOUNT As TransactionNestLevel     --To će se vratiti na jedan
                                               --na ovom mestu
ROLLBACK TRAN
SELECT @@TRANCOUNT As TransactionNestLevel     --I ponovo nula
                                               --na ovom mestu
```

Znate da bi u ovom primeru, vrednost `@@TRANCOUNT` na kraju takođe bila nula da je poslednja naredba bila `COMMIT`.

FUNKCIJE ZA TEKST I SLIKE

Funkcije za tekst i slike vrše operacije nad tekstualnim ili slikovnim podacima. To su:

- `TEXTPTR`
- `TEXTVALID`

TEXTPTR

Funkcija `TEXTPTR` proverava da li postoji pokazivač za tekst koji odgovara koloni tipa `text`, `ntext` ili `image` i vraća vrednost tipa `varbinary`. Pokazivač za tekst treba da se proveri kako bi se obezbedilo da pokazuje prvu tekstualnu stranicu da bi mogle da se izvrše naredbe `READTEXT`, `WRITETEXT`, ili `UPDATE`. Sintaksa je sledeća:

```
TEXTPTR(<column>)
```

TEXTVALID

Funkcija `TEXTVALID` proverava da li je navedeni pokazivač za tekst valjan. Sintaksa je sledeća:

```
TEXTVALID('<table.column>', <text_pointer>)
```

Parametar `table.column` je ime tabele i kolone koja će se koristiti. Parametar `text_pointer` je pokazivač za tekst koji treba da se proveri.

Ova funkcija će vratiti `0` ako pokazivač nije valjan, a `1` ako jeste.