მონაცემთა ბაზები

შეხვედრა 3

მონაცემთა ბაზების არქიტექტურა

სერვერზე მონაცემები მონაცემთა ბაზებში ინახება. არსებობს მონაცემთა ბაზის ორი სტრუქტურა: ლოგიკური და ფიზიკური.

მონაცემთა ბაზის **ლოგიკური სტრუქტურა** მოიცავს: ცხრილების სტრუქტურას, მათ შორის კავშირებს, მომხმარებლების სიას, შენახულ პროცედურებს და მონაცემთა ბაზის სხვა ობიექტებს.

მონაცემთა ბაზის **ფიზიკური სტრუქტურა** მოიცავს: მონაცემთა ბაზის ფაილებისა და ტრანზაქციების ჟურნალის აღწერას, მათ საწყის ზომას, ნაზარდს ზომას, მაქსიმალურ ზომას, კონფიგურირების პარამეტრებს და სხვა.

მონაცემთა ბაზების ლოგიკური სტრუქტურა:

მონაცემთა ბაზების ობიექტები - 1

მონაცემთა ბაზების ობიექტებია:

- **ცხრილები (Tables)** - ორგანზომილებიანი მატრიცებია, რომლებშიც უშუალოდ მონაცემები ინახება;
- **მონაცემთა ტიპები (Data Types)** - სისტემური მონაცემთა ტიპებია;
- **პირველადი (Primary Key) და გარე (Foreign Key) გასაღებები** - მონაცემთა ცხრილებს შორის კავშირების განსახორციელებელი სვეტები;
- **ცხრილებს შორის კავშირები** - მთავარი ცხრილის მონაცემების დამოკიდებულ ცხრილის მონაცემებთან კავშირში სრული ინფორმაციის სანახავად;
- **ნაგულისხმევი მნიშვნელობა (Default Value or Binding)** - მონაცემთა ბაზის ობიექტებია, რომლებიც შეგვიძლია მივუთითოთ უშუალოდ ცხრილის სტრუქტურაში ან მომხმარებლების მიერ განსაზღვრულ ტიპებში;
- **თვლადი სვეტი (Computed Column)** - მონაცემთა ცხრილის სტრიქონის სვეტში ამავე სტრიქონის სხვა სვეტებისაგან შემდგარი მათემატიკური მოქმედების ჩაწერა;

მონაცემთა ბაზების ობიექტები - 2

ასევე მონაცემთა ბაზების ობიექტებია:

- **წარმოდგენები (Views)** - ვირტუალური ცხრილებია, რომლებიც საშუალებას გვაძლევს ამორჩევის შედეგთან ვიმუშაოთ როგორც ცხრილთან;
- **ინდექსები (Indexes)** - სტრუქტურებია, რომლებიც ზრდის მონაცემებთან მუშაობის მწარმოებლურობას;
- **ტრიგერები (Triggers)** - სპეციალური შენახული პროცედურებია, რომლებიც ცხრილში მონაცემების შეცვლის დროს გამოიძახება;
- **მომხმარებლების ფუნქციები (user-defined functions)** - მომხმარებლების მიერ შექმნილი ფუნქციებია;
- **შენახული პროცედურები (stored procedures)** - Transact_SQL-ის ბრძანებების ნაკრებია, შენახული გარკვეული სახელით. მათთან მიმართვა შესაძლებელია სახელის საშუალებით;
- **მთლიანობის შეზღუდვები (constraints)** - ობიექტებია, რომლებიც უზრუნველყოფენ მონაცემების ლოგიკურ მთლიანობას და არ არსებობს ცხრილებისაგან დამოუკიდებლად.

ცხრილები

მონაცემთა ბაზის ობიექტებიდან მხოლოდ ცხრილი შეიცავს საკუთრივ მონაცემებს. ცხრილი შედგება სტრიქონებისა და სვეტებისაგან:

- თითოეული **სტრიქონი (row)** წარმოადგენს კონკრეტული ობიექტის მახასიათებლების ერთობლიობას. (მაგ., სტუდენტისათვის არის: გვარი, სახელი, უნივერსიტეტის დასახელება, კურსი და სხვ.)
- თითოეული **სვეტი (column)** წარმოადგენს ობიექტის მახასიათებელს ან მახასიათებლების ერთობლიობას. (მაგ., გვარი, სახელი და სხვ.). სვეტი არის ცხრილის მინიმალური ელემენტი. ცხრილის თითოეულ სვეტს აქვს სახელი, ტიპი და ზომა და სხვა მახასიათებლები.

ცხრილის ზოგიერთი სვეტი შეიძლება იყოს გამოთვლადი (computed). ასეთ სვეტებში ეთითება ფორმულა, რომლის მიხედვით სვეტის მნიშვნელობა გამოითვლება.

სისტემური ცხრილები

სისტემური ცხრილები (system tables) შეიცავენ სერვერის მუშაობისათვის საჭირო ინფორმაციას. ამიტომ, უფლება არ გვაქვს შეცვალოთ ამ ცხრილებში მოთავსებული მონაცემები.

აქედან გამომდინარე, ამ ცხრილების მიმართ შეგვიძლია მხოლოდ SELECT ბრძანების გამოყენება.

P.S. თუმცა აქვე უნდა აღვნიშნოთ, რომ შენახული პროცედურების გამოყენებით შეგვიძლია განვახორციელოთ მონაცემების ცვლილებები სისტემურ ცხრილებში.

დროებითი ცხრილები

დროებითი ცხრილები (temporary tables) განკუთვნილია მონაცემების დროებით შესანახად და მოთავსებულია tempdb სისტემურ მონაცემთა ბაზაში. დროებითი ცხრილები ავტომატურად იშლება შეერთების დახურვის ან სერვერის გაჩერების შემთხვევაში. არსებობს ორი სახის დროებითი ცხრილი:

- **ლოკალური დროებითი ცხრილი (local temporary tables)**, რომლის სახელი ერთი # სიმბოლოთი იწყება (მაგ., #LDC) და ამ სიმბოლოს მითითება საკმარისია ლოკალური დროებითი ცხრილის შესაქმნელად. ასეთი ცხრილი ხილულია მხოლოდ იმ შეერთების შიგნით, რომელშიც ის იქმნება, ხოლო შეერთების დახურვისას ის ავტომატურად იშლება. თუ ლოკალური დროებითი ცხრილი შეიქმნა შენახული პროცედურის მუშაობისას, მაშინ ამ პროცედურიდან გამოსვლისას ის ავტომატურად წაიშლება.
- **გლობალური დროებითი ცხრილი (global temporary table)**, რომლის სახელი ## სიმბოლოებით იწყება (მაგ., ##GDC) და ორჯერ ამ სიმბოლოს მითითება საკმარისია გლობალური დროებითი ცხრილის შესაქმნელად. ასეთი ცხრილი ხილულია ნებისმიერი შეერთებიდან და განკუთვნილია მონაცემების გასაცვლელად სხვადასხვა პროგრამას შორის. გლობალური დროებითი ცხრილი არსებობს იმ შეერთების დასრულებამდე, რომელშიც ის შეიქმნა, ხოლო მისი წაშლა და შეცვლა შეგვიძლია ყველა შეერთებიდან.

მონაცემთა ტიპები - 1

ცხრილის ყოველ სვეტს გააჩნია განსაზღვრული ტიპი, რომელიც მიუთითებს თუ როგორი ტიპის ინფორმაცია იქნება მოთავსებული ამ სვეტში. განვიხილოთ SQL-ის ძირითადი მონაცემთა ტიპები:

- **BINARY(N)** – ფიქსირებული N სიგრძის ორობითი მონაცემი (მაქსიმუმ 8000 ბაიტამდე);
- **BIGINT** - მთელი რიცხვი დიაპაზონში $-2^{63} \div 2^{63}-1$;
- **BIT** - მნიშვნელობა 0 ან 1;
- **CHAR(N)** - ფიქსირებული N სიგრძის ტექსტი (მაქსიმუმ 8000 სიმბოლომდე);
- **DATE** - თარიღი 0001 წლის 01 იანვრიდან 9999 წლის 31 დეკემბრამდე;
- **DATETIME** - თარიღი და დრო 1753 წლის 1 იანვრიდან 9999 წლის 31 დეკემბრამდე;
- **DECIMAL** - წილადი დიაპაზონში $2^{38}+1 \div 2^{38}-1$;
- **FLOAT** - წილადი დიაპაზონში -1,79E+308 ÷ −1,79E+308;
- **IMAGE** - ორობითი მონაცემი (მაქსიმუმ 2 გიგაბაიტამდე);
- **INT** - მთელი რიცხვი დიაპაზონში $-2^{31} \div 2^{31}-1$;

მონაცემთა ტიპები - 2

- **MONEY** - ფულის ერთეული დიაპაზონში $-2^{63} \div 2^{63}-1$;
- **NCHAR(N)** - ფიქსირებული N სიგრძის Unicode ტექსტი (მაქსიმუმ 8000 სიმბოლომდე);
- **NTEXT** - ცვლადი სიგრძის Unicode ტექსტი (მაქსიმუმ 1 გიგაბაიტ სიმბოლომდე);
- **NVARCHAR(N)** - ცვლადი N-მდე სიგრძის Unicode ტექსტი (მაქსიმუმ 8000 სიმბოლომდე);
- **NUMERIC** - იგივეა, რაც DECIMAL;
- **REAL** - მცურავ წერტილიანი რიცხვი დიაპაზონში -3,40E+38 ÷ −3,40E+38;
- **SMALLDATETIME** - თარიღი და დრო 1900 წლის 1 იანვრიდან 2079 წლის 6 ივნისამდე;
- **SMALLINT** - მთელი რიცხვი დიაპაზონში $-2^{15} \div 2^{15}-1$;
- **SMALLMONEY** - ფულის ერთეული დიაპაზონში $-2^{31} \div 2^{31}-1$;
- **SQL_VARIANT** - შესაძლებელია სხვადასხვა ტიპის მონაცემი (ამ ტიპის სვეტში ერთდროულად შეგვიძლია შევინახოთ INT, DATETIME, FLOAT და NVARCHAR(N) ტიპის მონაცემები);

მონაცემთა ტიპები - 3

- **SYSNAME** - მომხმარებლის მიერ შექმნილი ტიპი;
- **TABLE** - ეს ტიპი გამოიყენება მონაცემთა ნაკრებების შესანახად (Transact-SQL ცვლადებისა და ფუნქციებიდან დაბრუნებული მნიშვნელობებისათვის). ის არ გამოიყენება სვეტების მიმართ;
- **TEXT** - ცვლადი სიგრძის ტექსტი, რომელიც არ შეიცავს Unicode სიმბოლოებს (მაქსიმუმ 2^{31}-1 სიმბოლომდე);
- **TIMESTAMP** - დროითი შტამპია, რომლის მნიშვნელობა ავტომატურად იცვლება სტრიქონის ცვლილებისას;
- **TINYINT** - მთელი რიცხვი დიაპაზონში 0 ÷ 255;
- **VARBINARY(N)** - ცვლადი N-მდე სიგრძის ორობითი მონაცემი (მაქსიმუმ 8000 ბაიტამდე);
- **VARCHAR(N)** - ცვლადი N-მდე სიგრძის ტექსტი, რომელიც არ შეიცავს Unicode სიმბოლოებს (მაქსიმუმ 8000 სიმბოლომდე);
- **UNIQUEIDENTIFIER** - გამოიყენება გლობალური უნიკალური იდენტიფიკატორების (Global Unique Identifier, GUID) შესანახად.

პირველადი გასაღები

ნორმალიზების მეორე ფორმიდან გამომდინარე, ნებისმიერ მონაცემთა ცხრილს გააჩნია ერთი ან მეტი სვეტი, რომლებიც ცალსახად განსაზღვრავენ თითოეულ სტრიქონს. ასეთ სვეტ(ებ)ს **პირველადი გასაღები (PRIMARY KEY)** ეწოდება, რომელიც უნდა აკმაყოფილებდეს შემდეგ მოთხოვნებს:

- **უნიკალურობა** - ცხრილში არ უნდა იყოს ორი ისეთი სტრიქონი, რომლებსაც პირველადი გასაღების ერთნაირი მნიშვნელობები აქვთ;
- **მინიმალურობა** - პირველად გასაღებში შემავალი სვეტებიდან ნებისმიერის გამოკლება იწვევს უნიკალურობის დარღვევას.

გასათვალისწინებელია რამდენიმე რჩევა:

- არ შეიძლება უნიკალური სვეტ(ებ)ის მნიშვნელობის შეცვლა, რადგან იგი გამოიწვევს უნიკალურობის რღვევას (მაგ., თუ უნიკალური სვეტი არის ,,გვარი", პიროვნების გვარის შეცვლის შემთხვევაში დაიკარგება ინფორმაცია როგორც მთავარ, ასევე დამოკიდებულ ცხრილებში);
- როდესაც პირველადი გასაღები შედგება რამდენიმე სვეტისაგან, უმჯობესია დამატებით შეიქმნას ერთი სვეტი და ის განისაზღვროს როგორც უნიკალური გასაღები.

გარე გასაღები

გარე გასაღები (FOREIGN KEY) იქმნება დამოკიდებულ ცხრილში, რომელიც მთავარი ცხრილს მიმართავს. იგი, ასევე, შეიძლება შედგებოდეს ერთი ან მეტი სვეტისაგან. გარე გასა-ღების მნიშვნელობა უნდა არსებობდეს მთავარ ცხრილში, ან უნდა იყოს განუსაზღვრელი. ასევე, მთავარი ცხრილის პირველადი გასაღების რომელიმე მნიშვნელობა შეიძლება არ იყოს დაკავშირებული დამოკიდებული ცხრილის არც ერთ სტრიქონის მნიშვნელობასთან.

ცხრილებს შორის კავშირის შექმნისას შესაძლებელია დაყენებულ იქნეს პირველადი გასაღების შეცვლა/წაშლისას მოქმედების ოთხი ვარიანტი:

- **უმოქმედოდ (No Action)** - მთავარ ცხრილში პირველადი გასაღების მნიშვნელობების შეცვლა/წაშლისას დამოკიდებულ ცხრილში არაფერი არ შეიცვლება (კავშირი იკარგება);
- **კასკადური (Cascade)** - მთავარ ცხრილში პირველადი გასაღების მნიშვნელობების შეცვლა/წაშლისას იგივე განხორციელდება დამოკიდებულ ცხრილის იგივე მნიშვნელობების მქონე სტრიქონებში (კავშირი არ იკარგება);
- **განუსაზღვრელობა (Set Null)** - პირველადი გასაღების მნიშვნელობის შეცვლა/წაშლისას გარე გასაღები იღებს განუსაზღვრელ (NULL) მნიშვნელობას (კავშირი იკარგება).
- **შეთანხმებით (Set Default)** - დასაშვებია პირველადი გასაღების მხოლოდ იმ მნიშვნელობის შეცვლა/წაშლა, რომლებიც არ არის დაკავშირებული დამოკიდებული ცხრილის სტრიქონებთან (კავშირი არ იკარგება).

ცხრილებს შორის კავშირები

ნორმალიზების მესამე ფორმაზე დაყვანის შემდეგ, მთავარი ცხრილის მონაცემების სრული ინფორმაციის სანახავად საჭირო გახდა კავშირის განხორციელება დამოკიდებულ ცხრილთან, რისთვისაც გამოიყენება პირველადი და გარე გასაღებები. არსებობს სამი ტიპის კავშირი მონაცემთა ცხრილებს შორის:

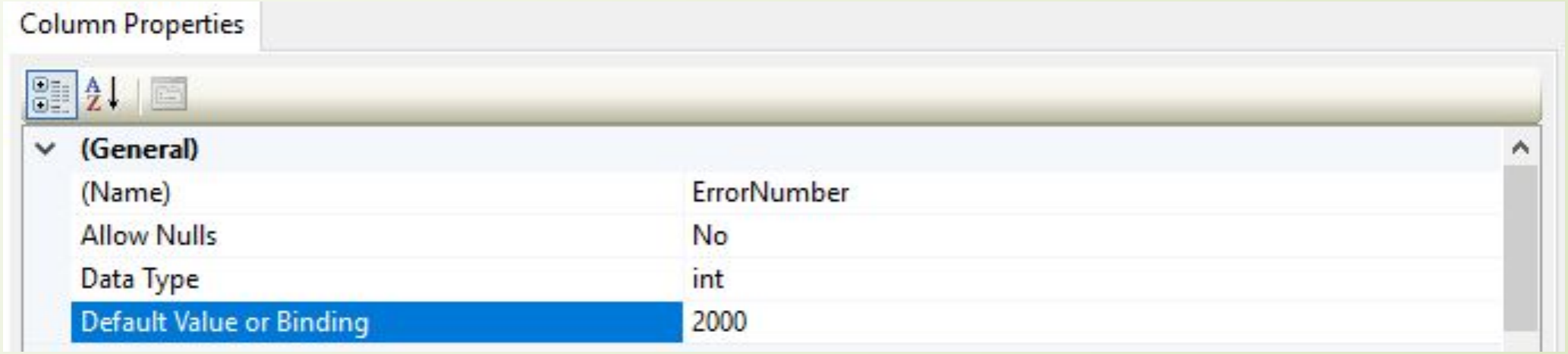
- **„ერთი-ერთთან“** - ერთი ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია მეორე ცხრილის ერთ ან არცერთ სტრიქონთან. ასეთი შემთხვევა წარმოიქმნება, როდესაც ერთი დიდი ცხრილი დაიყოფა ორ ან რამდენიმე ცხრილად და ყველა ცხრილში უნიკალური გასაღები იქნება მთავარი ცხრილის უნიკალური გასაღები;
- **„ერთი-ბევრთან“** - მთავარი ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია დამოკიდებული ცხრილის არცერთ, ერთ ან მეტ სტრიქონთან, ხოლო დამოკიდებული ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია მთავარი ცხრილის მხოლოდ ერთ სტრიქონთან. მონაცემთა ბაზებში უმეტეს შემთხვევაში ასეთი კავშირები წარმოიქმნება (მაგ., თანამშრომლები და თანამდებობები - ერთ თანამშრომელს მხოლოდ ერთი თანამდებობა აქვს, ხოლო ერთ თანამდებობაზე ბევრი თანამშრომლები მუშაობენ);
- **„ბევრი-ბევრთან“** - ერთი ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია მეორე ცხრილის არცერთ, ერთ ან მეტ სტრიქონთან და პირიქითაც იგივეა (მაგ., როგორც ერთ ავტორს აქვს რამდენიმე წიგნი, ასევე ერთ წიგნს ყავს რამდენიმე ავტორი).

ნაგულისხმევი მნიშვნელობა

ზოგჯერ საჭიროა მონაცემთა ცხრილში სტრიქონის დამატებისას გარკვეულ სვეტში თუ არ არის ცხადად მითითებული მნიშვნელობა ავტომატურად ჩაისვას მნიშვნელობა, რომელსაც **ნაგულისხმევი მნიშვნელობა (Default Value or Binding)** ეწოდება.

ნაგულისხმევი მნიშვნელობა შეიძლება იყოს მუდმივა, ჩადგმული ფუნქციის ან მათემატიკური გამოსახულების მიერ გადაცემული შედეგი.

გასათვალისწინებელია, რომ ნაგულისხმევი მნიშვნელობა არ უნდა იყოს კონფლიქტში მთლიანობის შეზღუდვებთან (მაგ., თუ ხელფასის სვეტისთვის მთლიანობის შეზღუდვაში განსაზღვრულია, რომ არ უნდა იყოს 2000-ზე მეტი, ნაგულისხმევი მნიშვნელობაც არ უნდა იყოს 2000-ზე მეტი).



სვეტი-მთვლელის განსაზღვრა

ზოგ მონაცემთა ცხრილს ვერ უთითებთ პირველად გასაღებს, რადგან არ გააჩნია თავისი უნიკალური მონაცემი და ამ ცხრილთან მუშაობისას უნდა განხორციელდეს სვეტი-მთვლელის დამატება და განსაზღვრა, რომლისთვისაც შესრულდება უნიკალური მნიშვნელობების ავტომატურად გენერირება IDENTITY საკვანძო სიტყვის გამოყენებით. სვეტი-მთვლელის განსაზღვრისას უნდა მივუთითოთ მისი საწყისი მნიშვნელობა და ბიჯი (მითითების გარეშე სისტემა შეთანხმებით ორივეს მნიშვნელობას 1-ის ტოლს იღებს. შედეგად, ცხრილში სტრიქონის ჩამატებისას შესაბამისი სვეტის მნიშვნელობა დაიწყება 1-ით და ყოველ დამატებაზე გაიზრდება 1-ით. ცხრილს შეიძლება ჰქონდეს მხოლოდ ერთი მთვლელი სვეტი.

სვეტისთვის IDENTITY თვისების განსაზღვრისას შემდეგი მოთხოვნები უნდა დავიცვათ:

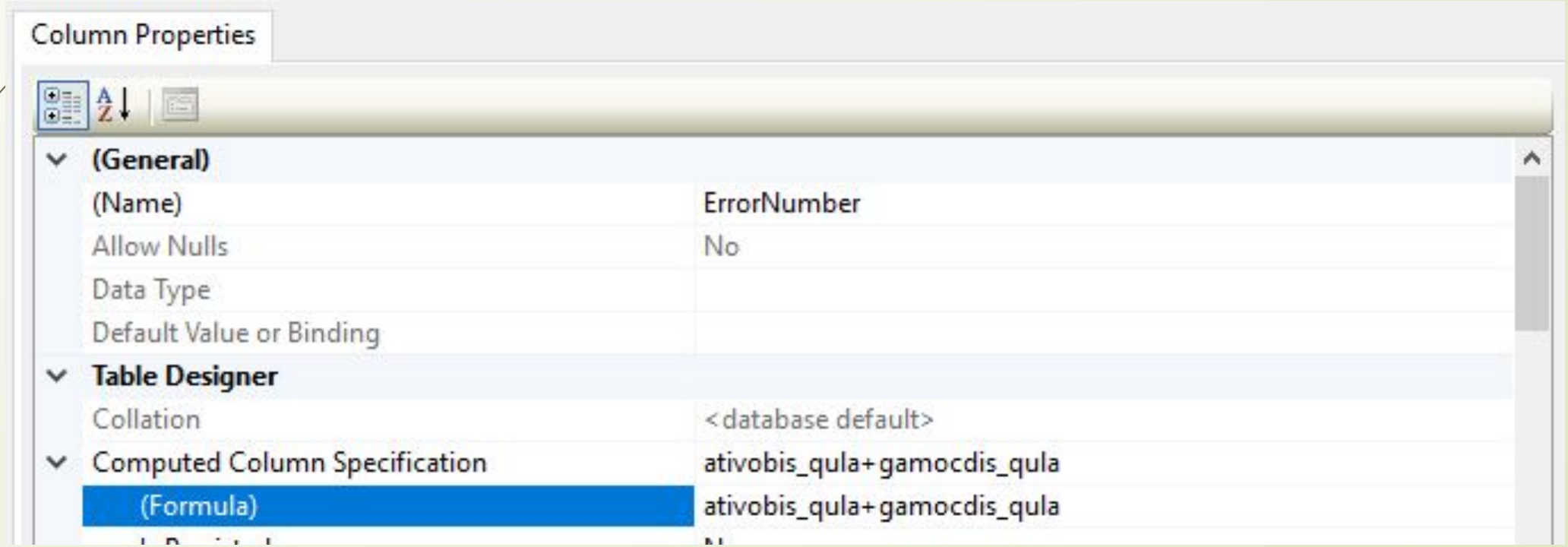
- სვეტის ტიპი უნდა იყოს: BIGINT, DECIMAL, INT, NUMERIC, SMALLINT ან TINYINT;
- სვეტისთვის აკრძალული უნდა იყოს NULL მნიშვნელობის შენახვა;
- სვეტისთვის არ უნდა იყოს განსაზღვრული ავტომატური მნიშვნელობა (ნაგულისხმევი, თვლადი, ფუნქცია და სხვ.).

მიმდინარე IDENTITY მნიშვნელობა (ანუ ცხრილში უკანასკნელად ჩასმული მნიშვნელობა) ინახება ამავე მონაცემთა ბაზის sys.identity_columns სისტემურ წარმოდგენაში.

თვლადი სვეტი

ზოგჯერ საჭიროა მონაცემთა ცხრილის სტრიქონის გარკვეულ სვეტში განხორციელდეს ამავე სტრიქონის სხვა სვეტებისაგან შემდგარი მათემატიკური მოქმედების ჩაწერა. ასეთ სვეტს **თვლადი სვეტი (Computed Column)** ეწოდება.

გასათვალისწინებელია, რომ თვლადი სვეტის ფორმულაში შეიძლება გამოყენებულ იქნეს მხოლოდ მათემატიკური მოქმედებები.



წარმოდგენები

წარმოდგენები (Views) არის ვირტუალური ცხრილი, რომელიც შედგება სხვა ცხრილ(ებ)ის ან/და წარმოდგენებისაგან და იგი განისაზღვრება SELECT მოთხოვნით. წარმოდგენა მონაცემებს არ შეიცავს - იგი გამოსახავს შემავალი ცხრილ(ებ)ის მონაცემებს. მონაცემები, რომლებიც გამოისახება ეკრანზე წარმოდგენის საშუალებით, ცალკე არ ინახება მონაცემთა ბაზაში.

მარტივი წარმოდგენა იქმნება ერთი ცხრილის ბაზაზე და შეიძლება შეიცავდეს სვეტების ფილტრებს და სორტირებებს, ხოლო რთული წარმოდგენა შედგება რამდენიმე დაკავშირებული ცხრილის ან/და წარმოდგენის საფუძველზე.

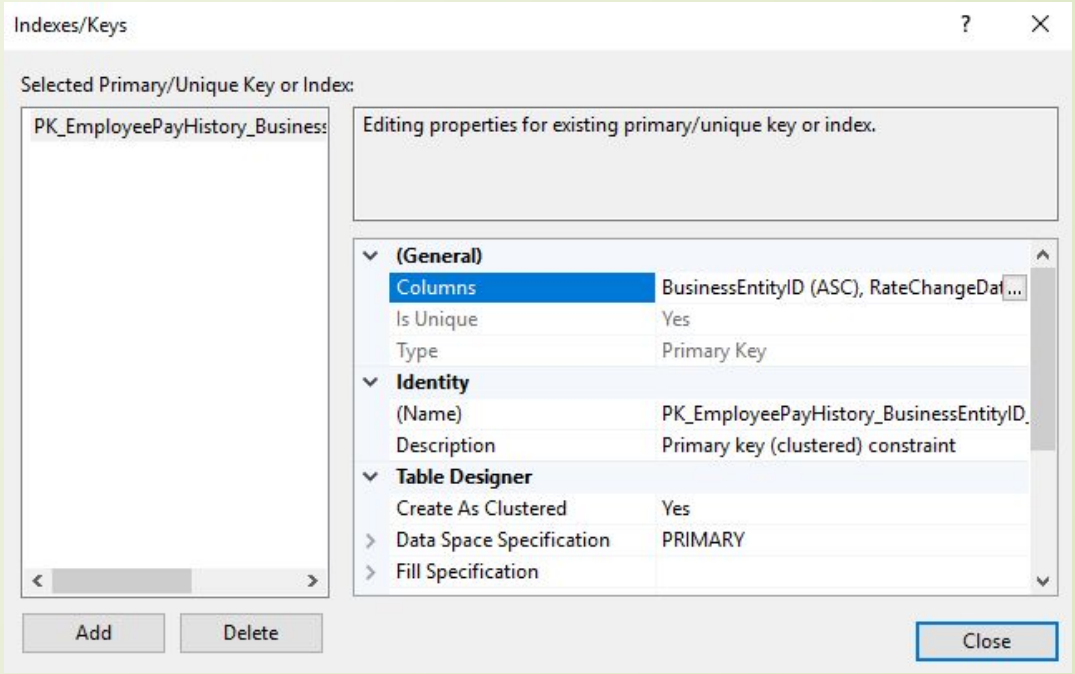
წარმოდგენასთან მიმართვა ხორციელდება ცხრილის ანალოგიურად სახელის გამოყენებით. წარმოდგენა გამოიყენება შემდეგ შემთხვევებში:

- მონაცემთა ცხრილ(ებ)ის გარკვეულ სტრიქონებთან/სვეტებთან მომხმარებლების მიმართვის შესაზღუდად;
- მონაცემთა დაკავშირებული ცხრილების მონაცემების ერთი ობიექტის სახით წარმოსადგენად;
- ინფორმაციის სანახავად, რომელიც მიიღება მონაცემების გარდაქმნის შედეგად.

ინდექსები

ინდექსები (Indexes) არის ცხრილთან ან წარმოდგენასთან დაკავშირებული სტრუქტურა და განკუთვნილია შესაბამის ცხრილში ან წარმოდგენაში ინფორმაციის ძებნის დასაჩქარებლად. ინდექსი განისაზღვრება ერთი ან მეტი სვეტისთვის, რომლებსაც ინდექსირებული სვეტები ეწოდებათ. ინდექსი შეიცავს ინდექსირებული სვეტის ან სვეტების დახარისხებულ მნიშვნელობებს საწყისი ცხრილის ან წარმოდგენის შესაბამის სტრიქონზე მიმართვებთან ერთად.

მწარმოებლურობის ზრდა მიიღწევა სვეტის მონაცემების დახარისხების ხარჯზე. თუ სვეტი დაუხარისხებელია, მაშინ საჭირო მნიშვნელობის საპოვნელად მიმდევრობით უნდა გაისინჯოს ყველა მნიშვნელობა. როცა საჭიროა ინდექსირებულ სვეტში მნიშვნელობის პოვნა, მაშინ მისი ძებნა სრულდება ინდექსში. ინდექსების გამოყენება მნიშვნელოვნად ზრდის მონაცემების ძებნის მწარმოებლურობას, მაგრამ ითხოვს დამატებით სივრცეს მონაცემთა ბაზაში.



მომხმარებლის ფუნქციები

ფუნქცია არის კონსტრუქცია, რომელიც შეიცავს ხშირად გამოყენებად კოდს და გააჩნია სახელი. ფუნქცია მონაცემებზე გარკვეულ მოქმედებებს ასრულებს და აბრუნებს კონკრეტულ წინასწარ განსაზღვრულ მნიშვნელობას. მისი გამოძახება მონაცემთა ცხრილისა და წარმოდგენის ანალოგიურად სახელით ხორციელდება ფრჩხილებში პარამეტრების მითითებით.

არსებობს მომხმარებლის ფუნქციების ორი ტიპი:

1. ჩადგმული ფუნქციები (Built-in Functions) - პროგრამირების გარემოს შემადგენელი ნაწილია და წინასწარ განსაზღვრულ მოქმედებებს ასრულებს (მაგ., MAX(), SUM() და სხვ.);
2. მომხმარებლის ფუნქციები (User-Defined Functions) - მომხმარებლის მიერ საჭიროებისამებრ შექმნილი კოდი თავისი სახელით.

ტრიგერები

ტრიგერები (Triggers) არის შენახული პროცედურის სპეციალური კლასი, რომლებიც ავტომატურად გაიშვება ცხრილებში მონაცემების დამატების, შეცვლის ან წაშლის დროს. ტრიგერები იყოფიან სამ კატეგორიად:

1. ჩამატების ტრიგერები (INSERT TRIGGER) - მუშაობას იწყებს მონაცემთა ცხრილში ახალი სტრიქონის დამატების შემდეგ;
2. შეცვლის ტრიგერები (UPDATE TRIGGER) - მუშაობას იწყებს მონაცემთა ცხრილში არსებული სტრიქონის ჩასწორების შემდეგ;
3. წაშლის ტრიგერები (DELETE TRIGGER) - მუშაობას იწყებს მონაცემთა ცხრილში არსებული სტრიქონის წაშლის შემდეგ.

ერთ ტრიგერში შესაძლებელია იყოს როგორც ცალ-ცალკე ჩამატების, შეცვლისა და წაშლის, ასევე მათი კომბინაციის მქონეც.

ერთი ცხრილისთვის შესაძლებელია განსაზღვრული იყოს თითოეული ტიპის რამდენიმე ტრიგერი. გარდა ამისა, ერთი ტრიგერიდან შესაძლებელია სხვა ტრიგერების გამოძახება - ასეთ შემთხვევაში გვაქვს ჩადგმული ტრიგერები (nested triggers).

შენახული პროცედურები

შენახული პროცედურები (Stored Procedures) არის ერთ მოდულში გაერთიანებული ბრძანებების ჯგუფი, რომელსაც აქვს სახელი. ეს ბრძანებები კომპილირდება და სრულდება, როგორც ერთი მთლიანი.

სერვერზე ინახება სისტემური შენახული პროცედურების დიდი რაოდენობა. მათი სახელები იწყება „sp_“ პრეფიქსით. ისინი გამოიყენება სერვერის კონფიგურირებისა და მართვისათვის, სისტემურ მონაცემთა ბაზებსა და ცხრილებში მონაცემების შესაცვლელად და ა.შ.

შესაძლებელია საკუთარი შენახული პროცედურის შექმნა, რომელიც მოთავსებული იქნება არსებულ მონაცემთა ბაზაში. შენახული პროცედურის შესასრულებლად გამოიყენება მისი სახელი და საჭიროების შემთხვევაში მიეთითება პარამეტრები.

როცა შენახული პროცედურა პირველად სრულდება, სერვერი ქმნის მისი შესრულების გეგმას. შენახული პროცედურის განმეორებით შესრულებისას, სერვერი იყენებს მეხსიერებაში მოთავსებულ გეგმას, რაც ზრდის მის წარმადობას.

ძირითად, შენახული პროცედურები გამოიყენება მონაცემთა დამუშავების შედეგად მიღებული ინფორმაციის ამოსაღებად მონაცემთა ბაზიდან.

მთლიანობის შეზღუდვები

მთლიანობის შეზღუდვები (Constraints) უზრუნველყოფენ ავტომატურ კონტროლს ჩვენ მიერ განსაზღვრულ პირობების/შეზღუდვების მონაცემების შესაბამისობაში ლოგიკურ დონეზე.

მთლიანობის შეზღუდვები შეიძლება გამოყენებული იყოს სვეტის ან ცხრილის დონეზე. მთლიანობის შეზღუდვები, რომლებიც სვეტის დონეზე გამოიყენება, მოქმედებს მხოლოდ ამ სვეტში შესატან მონაცემებზე. თუ მთლიანობის შეზღუდვა გამოიყენება რამდენიმე სვეტის მიმართ, მაშინ შეზღუდვა მუშაობს ცხრილის დონეზე.

ფუნქციურობის და გამოყენებით არსებობს მთლიანობის შეზღუდვის ხუთი ტიპი:

1. NULL (უცნობი) მთლიანობის შეზღუდვა. ის მოქმედებს სვეტისა და მომხმარებლის ტიპის დონეზე. მათთვის შეგვიძლია განვსაზღვროთ NULL ან NOT NULL მთლიანობის შეზღუდვა. შესაბამისად, სვეტში ნებადართული ან აკრძალული იქნება NULL მნიშვნელობის შენახვა. NULL არის სპეციალური მნიშვნელობა, რომელიც აღნიშნავს მნიშვნელობის არყოფნას. NULL არ არის იგივე, რაც ინტერვალის სიმბოლო, ნული ან ნულოვანი სიგრძის სტრიქონი. NULL მნიშვნელობა შეიძლება შეფასდეს როგორც UNKNOWN (უცნობი) და არ შეიძლება შეფასდეს როგორც TRUE ან FALSE;

მთლიანობის შეზღუდვები

2. CHECK (შემოწმება) მთლიანობის შეზღუდვა მოქმედებს სვეტის დონეზე და ზღუდავს სვეტში შესანახი მნიშვნელობების დიაპაზონს. მისი განსაზღვრისას შესატანი მონაცემებისათვის ეთითება ლოგიკური პირობა. მონაცემების შეტანის ან ჩამატების დროს შესატანი მნიშვნელობა მოთავსდება პირობაში. თუ შემოწმების შედეგია TRUE (ჭეშმარიტი), მაშინ მონაცემების შეცვლა ნებადართული იქნება. თუ შემოწმების შედეგია FALSE (მცდარი), მაშინ ცვლილებები აიკრძალება და გაიცემა შეტყობინება შეცდომის შესახებ. ერთი სვეტისთვის შეგვიძლია შევქმნათ რამდენიმე CHECK შეზღუდვა. მისი მითითება შეგვიძლია ცხრილის შექმნის დროს;
3. UNIQUE (უნიკალურობა) მთლიანობის შეზღუდვა მოქმედებს სვეტის დონეზე და იძლევა სვეტში მნიშვნელობების უნიკალურობის გარანტიას. ცხრილში არ იქნება ორი სტრიქონი, რომლებსაც მოცემულ სვეტში ერთნაირი მნიშვნელობები ექნებათ. პირველადი გასაღებისაგან განსხვავებით UNIQUE მთლიანობის შეზღუდვა უშვებს NULL მნიშვნელობის არსებობას;

მთლიანობის შეზღუდვები

4. PRIMARY KEY (პირველადი გასაღები) მოქმედებს სვეტის ან ცხრილის დონეზე. პირველადი გასაღები შეიძლება შედგებოდეს ერთი ან მეტი სვეტისაგან და არის სტრიქონის უნიკალური იდენტიფიკატორი ცხრილის ფარგლებში. თუ პირველადი გასაღები ერთი სვეტისაგან შედგება, მაშინ ამ სვეტისთვის უნდა იყოს დაყენებული UNIQUE შეზღუდვა, სვეტში მნიშვნელობების უნიკალურობის გარანტირებისათვის. თუ პირველადი გასაღები რამდენიმე სვეტისაგან შედგება, მაშინ თითოეულ სვეტში მნიშვნელობები შეიძლება გამეორდეს ანუ არ იყოს უნიკალური. მაგრამ, უნიკალური უნდა იყოს ამ სვეტების მნიშვნელობების კომბინაცია თითოეული სტრიქონისათვის. პირველად გასაღებში შემავალი არც ერთი სვეტისათვის არ უნდა იყოს დაყენებული NULL შეზღუდვა. ცხრილში შეგვიძლია შევქმნათ მხოლოდ ერთი პირველადი გასაღები.
5. FOREIGN KEY (გარე გასაღები) იქმნება ცხრილის დონეზე. დამოკიდებული ცხრილის გარე გასაღები უკავშირდება მთავარი ცხრილის პირველად გასაღებს. დამოკიდებულ ცხრილში არ შეიძლება სტრიქონის ჩასმა, თუ გარე გასაღებს არ აქვს შესაბამისი მნიშვნელობა მთავარ ცხრილში. მთავარი ცხრილიდან შეუძლებელია სტრიქონის წაშლა, თუ მასთან დაკავშირებულია თუნდაც ერთი სტრიქონი დამოკიდებულ ცხრილში. მთავარი ცხრილიდან სტრიქონის წაშლის წინ აუცილებელია წინასწარ წავშალოთ დამოკიდებული ცხრილის ყველა სტრიქონი.

სინტაქსის წაკითხვის წესები

T-SQL ენის თითოეულ ბრძანებას აქვს საკუთარი **სინტაქსი** ანუ სწორად დაწერის წესი. განვიხილოთ CREATE VIEW ბრძანების სინტაქსი, რომელიც წარმოდგენის შესაქმნელად გამოიყენება:

CREATE VIEW [სქემის_სახელი.] წარმოდგენის_სახელი
[(სვეტის_სახელი [, . . . n])]
[WITH <წარმოდგენის_ატრიბუტები> [, . . . n]]
AS
SELECT-ბრძანება [WITH CHECK OPTION]
<წარმოდგენის_ატრიბუტები> ::= { ENCRYPTION | SCHEMABINDING | VIEW_METADATA }

- ყოველგვარი ფრჩხილების გარეშე პირდაპირი ტექსტი - აუცილებელი არგუმენტი;
- კვადრატულ ფრჩხილებში (“[” და “]”) - არა აუცილებელი არგუმენტი;
- კუთხურ ფრჩხილებში (“<” და “>”) - კონსტრუქცია, რომელსაც თავისი სინტაქსი აქვს;
- ფიგურულ ფრჩხილებში (“{” და “}”) მოთავსებული არგუმენტებიდან ერთ-ერთი უნდა ავირჩიოთ, ხოლო “ | ” ნიშნავს "ან" ოპერატორს;
- [, . . . n] - არგუმენტის გამეორება, რომლებიც მძიმეთი იქნებიან გამოყოფილნი.

სისტემური მონაცემთა ბაზები

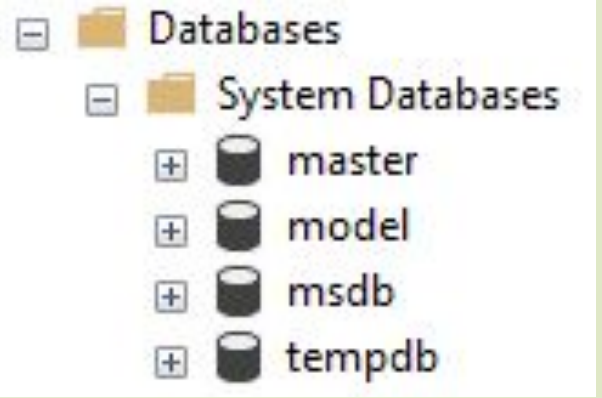
მონაცემთა ბაზების სერვერს გააჩნია ოთხი სისტემური მონაცემთა ბაზა:

- master;
- model;
- msdb;
- tempdb.

ეს მონაცემთა ბაზები გენერირდებიან სერვერის ინსტალაციისას, ინახავენ სისტემურ მონაცემებს და ემსახურებიან შიგა ამოცანების გადაწყვეტას.

როგორც სისტემური, ისე მომხმარებლების მიერ შექმნილი მონაცემთა ბაზები, შეიცავენ სისტემურ ცხრილებსა და წარმოდგენებს. ასევე, ისინი შეიცავენ ინფორმაციას მონაცემთა ბაზის სტრუქტურაზე და მის ორგანიზებულობაზე.

სისტემურ ცხრილებთან მუშაობა UPDATE, INSERT და DELETE ბრძანებების საშუალებით აკრძალულია. დასაშვებია SELECT ბრძანების გამოყენება. ამ ცხრილებში მონაცემების შეცვლა შესაძლებელია მხოლოდ სისტემური შენახული პროცედურების გამოყენებით.



master სისტემური მონაცემთა ბაზა

master მონაცემთა ბაზა შეიცავს:

- სრულ სისტემურ ინფორმაციას სერვერზე;
- ინფორმაციას სერვერის სხვა მონაცემთა ბაზებზე;
- ინფორმაციას სერვერის პირველადი ფაილების ადგილმდებარეობის შესახებ.

master მონაცემთა ბაზა შემდეგი ფაილებისაგან შედგება:

- master.mdf - მონაცემების ფაილი;
- mastlog.ldf - ტრანზაქციების ჟურნალის ფაილი.

ორივე ფაილი ინახება \Data კატალოგში (C:\Program Files\Microsoft SQL Server\...\MSSQL\Data\).

დაუშვებელია master სისტემურ მონაცემთა ბაზაში მომხმარებლის ობიექტების შექმნა.

model სისტემური მონაცემთა ბაზა

სერვერზე ახალი მონაცემთა ბაზის შექმნა ხორციელდება მასში model სისტემური მონაცემთა ბაზის ობიექტების გადაწერის გზით.

model მონაცემთა ბაზა მოთავსებულია \Data კატალოგში და ორი ფაილისგან შედგება:

- model.mdf - მონაცემების ფაილი;
- model.ldf - ტრანზაქციების ჟურნალი.

msdb სისტემური მონაცემთა ბაზა

msdb სისტემური მონაცემთა ბაზა გამოიყენება SQL Server Agent სამსახურის მიერ მოვლენების (alerts), ამოცანებისა (jobs) და ოპერატორების (operators) რეგისტრირების დაგეგმვისათვის.

msdb მონაცემთა ბაზა ინახავს მთელ ინფორმაციას, რომელიც ეხება ადმინისტრირების ავტომატიზებასა და სერვერის მართვას.

msdb ბაზა ორი ფაილისგან შედგება:

- msdbdata.mdf - მონაცემების ფაილი;
- msdblog.ldf - ტრანზაქციების ჟურნალი.

tempdb სისტემური მონაცემთა ბაზა

tempdb მონაცემთა ბაზა შეიცავს დროებით ობიექტებს, როგორიცაა ცხრილები, შენახული პროცედურები, ცვლადები, კურსორები და ა.შ. მასში, ინახება, როგორც სისტემური, ისე მომხმარებლის მიერ შექმნილი ობიექტები. სერვერის ყოველი გაშვების დროს tempdb მონაცემთა ბაზა ხელახლა იქმნება და ყოველთვის იშლება სერვერის გაჩერების დროს. მონაცემთა ბაზის ობიექტები ინახება მხოლოდ ერთი სეანსის განმავლობაში.

tempdb მონაცემთა ბაზა არის გლობალური კურსორი, რომელიც ავტომატურად მისაწვდომია ყველა მომხმარებლისათვის. შესაქმნელი დროებითი ობიექტები შეიძლება იყოს როგორც ლოკალური, ისე გლობალური. ლოკალური ობიექტი მისაწვდომია მხოლოდ მისი შემქმნელი მომხმარებლისთვის, გლობალური. კი - ყველა მომხმარებლისათვის. ლოკალური ობიექტები მოქმედებენ მხოლოდ სეანსის, შენახული პროცედურის, ტრიგერის ან ბრძანებების პაკეტის ფარგლებში. დროებითი ობიექტის შემქმნელი სტრუქტურიდან გამოსვლისას ეს ობიექტი მაშინვე წაიშლება, როგორც კი მომხმარებელი დაამთავრებს სერვერთან მუშაობას. ლოკალური დროებითი ცხრილის სახელი # სიმბოლოთი იწყება, გლობალური დროებითი ცხრილის სახელი კი - ## სიმბოლოებით.

tempdb მონაცემთა ბაზა ორი ფაილისაგან შედგება:

- tempdb.mdf - მონაცემების ფაილი;
- templogldf - ტრანზაქციების ჟურნალი.

ფაილები და ფაილების ჯგუფები - 1

მონაცემთა ბაზის შესანახად სამი ტიპის ფაილი გამოიყენება:

- Primary - ძირითადი ფაილია, რომელიც შეიცავს სისტემურ ინფორმაციას თვით მონაცემთა ბაზისა და მისი ობიექტების შესახებ. მასში მოთავსებულია სისტემური ცხრილები და მონაცემთა ბაზის ობიექტების აღწერა. ძირითად ფაილში შეიძლება, აგრეთვე ინახებოდეს მონაცემებიც. ნებისმიერ მონაცემთა ბაზას აქვს primary ტიპის ფაილი, ამასთან მონაცემთა ბაზაში ამ ტიპის ფაილი მხოლოდ ერთი შეიძლება იყოს. primary ტიპის ფაილებს აქვთ .mdf (Master Data File) გაფართოება.
- Secondary - მონაცემების არაძირითადი ფაილია, რომელიც არ შეიცავს სისტემურ ინფორმაციას და განკუთვნილია მხოლოდ მონაცემების შესანახად. მონაცემები, რომლებიც არ დაეტია ძირითად ფაილში, მოთავსდება მონაცემების არაძირითადი ფაილებში. მონაცემთა ბაზას შეიძლება საერთოდ არ ჰქონდეს secondary ტიპის ფაილი ან ჰქონდეს რამდენიმე. secondary ტიპის ფაილებს აქვთ .ndf (Not Master Data File) გაფართოება.
- Transaction Log - ტრანზაქციების ჟურნალის ფაილია. ის გამოიყენება მონაცემთა ბაზაში შესრულებული ტრანზაქციების შესახებ ინფორმაციის შესანახად. ნებისმიერ მონაცემთა ბაზას აქვს ტრანზაქციების ჟურნალის მინიმუმ ერთი ფაილი. transaction log ტიპის ფაილებს აქვთ .ldf (Log Data File) გაფართოება.

ფაილები და ფაილების ჯგუფები - 2

მონაცემთა ბაზაში გამოყენებულ თითოეულ ფაილს ორი სახელი აქვს:

1. ფაილის ლოგიკური სახელი (Logical File Name), რომელიც გამოიყენება Transact-SQL ბრძანებებში კონკრეტულ ფაილთან მიმართვისათვის.
2. ფაილის სახელი ოპერაციულ სისტემაში (OS File Name), რომელიც არის ფაილის ფიზიკური სახელი და ამ სახელით ინახება ფაილი დისკზე.

მონაცემთა ბაზის ყველა ფაილის ზომა შეიძლება ავტომატურად გაიზარდოს, რომელიც შესაძლებელია ფაილის შექმნის დროს განისაზღვროს. ნაზარდის ზომა შეგვიძლია განვსაზღვროთ პროცენტებში (ფაილის საწყისი ზომიდან) ან მეგაბაიტებში. დამატებით, შეგვიძლია მივუთითოთ ფაილის მაქსიმალური ზომა. თუ მაქსიმალური ზომა არ არის მითითებული, მაშინ ფაილის ზომა გაიზრდება მანამ, სანამ არის თავისუფალი სივრცე დისკზე.

ფაილები და ფაილების ჯგუფები - 3

ადმინისტრირებისა და მონაცემთა ბაზის ობიექტების ფიზიკური განლაგების მართვის გაადვილების მიზნით უმჯობესია ფაილები საფაილო ჯგუფებში მოვათავსოთ, რომლებიც არიან სამი ტიპის:

1. ფაილების ძირითადი ჯგუფი (Primary File Group) - შეიცავს primary ტიპის ფაილსა და ყველა ფაილს, რომლებიც არ არის ჩართული სხვა ჯგუფში (შეიძლება მხოლოდ ერთი ძირითადი ჯგუფი);
2. მომხმარებლების მიერ შექმნილი ფაილების ჯგუფი (User-defined File Group) - ჩართულია ყველა ფაილი, რომლებიც მითითებულია FILEGROUP პარამეტრში მონაცემთა ბაზის შექმნის (CREATE DATABASE) ან შეცვლის (ALTER DATABASE) დროს (შეიძლება რამდენიმე ჯგუფი ფაილების ნებისმიერი შემადგენლობით).
3. ფაილების ნაგულისხმევი ჯგუფი (Default File Group) - მონაცემთა ბაზაში ფაილების ერთ-ერთი ჯგუფი ხდება ნაგულისხმევი. თუ მონაცემთა ბაზის შექმნის დროს იგი არ არის მითითებული, მაშინ ფაილების ძირითადი ჯგუფი ხდება ნაგულისხმევი. თუ მონაცემთა ბაზის ობიექტის (ცხრილის ან სვეტის) შექმნის დროს აშკარად არ არის მითითებული, ფაილების რომელ ჯგუფს ეკუთვნის ის, მაშინ ეს ობიექტი შეიქმნება ფაილების ნაგულისხმევ ჯგუფში. მონაცემთა ბაზის მფლობელმა შეიძლება ფაილების ნებისმიერი ჯგუფი დანიშნოს ფაილების ნაგულისხმევ ჯგუფად. ფაილების მხოლოდ ერთი ჯგუფი შეიძლება იყოს ნაგულისხმევი.