Лабораторна робота № 1

Тема: Початкове проектування бази даних

Короткі теоретичні відомості:

В **SQL Server**'і бази даних зберігаються у вигляді звичайних файлів на диску. Як мінімум на одну БД доводиться таких файлів 2: ***.mdf** і ***.ldf**. У першому зберігаються самі дані, таблиці, індекси та ін., а в другому перебуває т.зв. **transaction log**, у якому перебуває інформація необхідна для відновлення БД.

Оператор **CREATE DATABASE** може повторити всі ваші дії, описані вище. Власне Enterprise Manager "перевів" ваші натискання в цей оператор і передав його SQL Server'у. Майже всі функції Enterprise Manager'а саме так і виконуються: у користувача запитуються дані, формується оператор чудової мови SQL (який ми розглянемо пізніше) і передається для виконання SQL Server, а результати виконання показують користувачеві.

Розглянемо ближче синтаксис оператора **CREATE DATABASE**.

```
CREATE DATABASE database_name
[ON
   { [PRIMARY] (NAME = logical_file_name,
      FILENAME = 'os_file_name'
      [, SIZE = size]
      [, MAXSIZE = max_size]
      [, FILEGROWTH = growth_increment] )
   } [,...n]
]
[LOG ON
   { ( NAME = logical_file_name,
      FILENAME = 'os_file_name'
      [, SIZE = size] )
   } [,...n]
]
[FOR RESTORE]
```

FILENAME - повний шлях і ім'я файлу для розміщення БД, повинен вказувати на локальний диск комп'ютера, на якому встановлений SQL Server.

SIZE - початковий розмір кожного файлу в Мб.

MAXSIZE - максимальний розмір файлу в Мб, якщо не зазначений розмір не обмежується.

FILEGROWTH - одиниця збільшення файлу, вказується в Мб (за замовчуванням) або у відсотках (тобто до числа додається %), значення 0 забороняє збільшення файлу.

Тепер подивимося як виглядає створення БД:

```
CREATE DATABASE SQlTest
ON
        PRIMARY (NAME=SQlTest,
        FILENAME='D:\MSSQL\data\SQlTest_data.mdf',
        SIZE=5,
        MAXSIZE=10,
        FILEGROWTH=10% )
LOG ON (
        NAME=SQLStepLog,
        FILENAME='D:\MSSQL\data\SQlTest_log.ldf',
        SIZE=1,
        MAXSIZE=5,
        FILEGROWTH=1
```

)

Обмеження цілісності (constraints) — це механізм, що забезпечує автоматичний контроль відповідності даних установленим умовам (або обмеженнями). Тобто обмеження цілісності використовуються для забезпечення цілісності даних на логічному рівні.

Є п'ять класів обмежень цілісності, що різняться по функціональності й області застосування.

- *Обмеження цілісності NULL.* Діє на рівні стовпця й користувальницького типу даних. Встановлюючи для них обмеження цілісності NULL або NOT NULL, можна відповідно дозволяти або забороняти зберігання значень NULL.
- *Обмеження цілісності CHECK (перевірка).* Це обмеження цілісності діє на рівні стовпця й обмежує діапазон значень, які можуть бути збережені в стовпці. При визначенні цього обмеження цілісності вказується логічна умова для даних, що вводяться. При введенні або зміні даних вводиться значення, що, підставляється в умову. Якщо в результаті перевірки вертається значення «істина» (TRUE), то зміни даних приймаються. Якщо ж перевірка повертає «неправда» (FALSE), то зміни даних відкидаються й генерується повідомлення про помилку. Для одного стовпця можна створити кілька обмежень цілісності CHECK.
- *Обмеження цілісності UNIQUE (унікальність).* Діє на рівні стовпця й гарантує унікальність значень у стовпці. У таблиці не може бути двох рядків, що мають однакове значення в стовпці із установленим обмеженням цілісності UNIQUE. На відміну від первинного ключа, обмеження цілісності UNIQUE допускає зберігання значень NULL.
- *Обмеження цілісності PRIMARY KEY (первинний ключ).* Діє на рівні стовпця або таблиці. Первинний ключ складається з одного або декількох стовпців, що унікально ідентифікують рядок у межах таблиці. Якщо для генерації первинного ключа використовується єдиний стовпець, то для цього стовпця повинне бути встановлене обмеження цілісності UNIQUE для гарантії унікальності значень у стовпці. Ні для одного зі стовпців, включених у первинний ключ, не повинне бути встановлене властивість NULL. В якості унікального ключа у таблиці може бути використаний один з декількох стовпців або їхніх комбінацій. Всі ці комбінації є *кандидатами* на первинний ключ (можливими ключами). Адміністратор повинен вибрати тільки одного кандидата й створити на його базі первинний ключ. Таким чином, у таблиці може бути створений тільки один первинний ключ.
- *Обмеження цілісності FOREIGN KEY (зовнішній ключ).* Створюється на рівні таблиці. Зовнішній ключ зв'язується з одним з *кандидатом* на первинний ключ в іншій таблиці. Таблиця, у якій визначений зовнішній ключ, називається *залежною*, а таблиця з кандидатом на первинний ключ - *головною*. У залежну таблицю не можна вставити рядок, якщо зовнішній ключ не має відповідного значення в головній таблиці. Крім того, з головної таблиці не можна видалити рядок, якщо з нею зв'язана хоча б один рядок у залежній таблиці. Перед видаленням рядка з головної таблиці необхідно перед цим видалити всі рядки із залежної таблиці. Для цього можна написати спеціальну збережену процедуру або створити тригер.

Правила (rule) - використовуються для обмеження значень, збережених у стовпці таблиці або в користувальницькому типі даних. Правила залишені для забезпечення зворотної сумісності з попередніми версіями SQL Server.

Правило може бути створено тільки в поточній базі даних. Створення правила виконується за допомогою команди CREATE RULE, тоді як для зв'язування правила з об'єктом бази даних використовується системна збережена процедури sp_bindru1e. Для видалення зв'язку між правилом і об'єктом бази даних застосовується збережена процедура sp_unbindrule.

Завдання:

Початкове проектування БД на основі універсального відношення:

- іменування полів і визначення їх типів;
- визначення обмежень (constraints: null/not null, default, check) для полів відношень;
- створення додаткових типів даних (user defined type, domain);
- формулювання додаткових ділових правил (business rules).

Оформлення сценарію sql що містить команди для створення БД, таблиці універсального відношення, завантаження даних у цю таблицю.