Вариант 01.

Создать хранимую процедуру без параметров, в которой для экземпляра SQL Server создаются резервные копии всех пользовательских баз данных. Имя файла резервной копии должно состоять из имени базы данных и даты создания резервной копии, разделенных символом нижнего подчеркивания. Дата создания резервной копии должна быть представлена в формате YYYYDDMM. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Для создания резервной копии базы данных воспользоваться инструкцией BACKUP DATABASE, имеющей следующий формат:

BACKUP DATABASE *имя_базы_данных* TO DISK = *полная_спецификация_файла_резервной_копии*

2. Имена баз данных экземпляра SQL Server можно извлечь из системного представления sys.databases.

Вариант 02.

Создать хранимую процедуру, которая, не уничтожая базу данных, уничтожает все те таблицы текущей базы данных в схеме 'dbo', имена которых начинаются с фразы 'TableName'. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Для уничтожения таблицы воспользоваться инструкцией DROP TABLE.
2. Информацию о таблицах базы данных можно извлечь из системного представления sys.objects.

Вариант 03.

Создать хранимую процедуру с входным параметром, которая выводит имена и описания типа объектов (только хранимых процедур и скалярных функций), в тексте которых на языке SQL встречается строка, задаваемая параметром процедуры. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию об объектах базы данных можно извлечь из системного представления sys.objects.
2. Информацию о модулях базы данных можно извлечь из системного представления sys.sql_modules.

Вариант 04.

Создать хранимую процедуру с двумя входными параметрами – имя базы данных и имя таблицы, которая выводит сведения об индексах указанной таблицы в указанной базе данных. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Сведения об индексах указанной таблицы в указанной базе данных можно получить с помощью динамической административной функции sys.dm_db_index_physical_stats, принимающей пять параметров:
    1. Идентификатор базы данных - можно получить с помощью функции метаданных DB_ID.
    2. Идентификатор таблицы - можно получить с помощью функции метаданных OBJECT_ID.
    3. Для прочих параметров можно указать значения NULL.
2. Если имя базы данных задано неверно, напечатать сообщение 'Invalid database'.
3. Если имя таблицы задано неверно, напечатать сообщение 'Invalid table'.

Вариант 05.

Создать хранимую процедуру с входным параметром – «имя таблицы», которая удаляет дубликаты записей из указанной таблицы в текущей базе данных. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Если имя таблицы задано неверно, напечатать сообщение 'Invalid table'.

Вариант 06.

Создать хранимую процедуру без параметров, которая перестраивает все индексы всех таблиц в схеме 'dbo' в текущей базе данных. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию о таблицах базы данных можно извлечь из системного представления sys.objects.
2. Для перестроения индексов для таблицы в указанной базе данных воспользоваться командой DBCC DBREINDEX.
3. Использовать значение коэффициента заполнения для индекса, указанное при создании индекса.

Вариант 07.

Создать хранимую процедуру без параметров, которая в текущей базе данных обновляет все статистики для таблиц в схеме 'dbo'. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию о таблицах базы данных можно извлечь из системного представления sys.objects.
2. Чтобы обновить статистику оптимизации запросов для таблицы в указанной базе данных, необходимо воспользоваться инструкцией UPDATE STATISTICS, которая в простейшем случае имеет следующий формат:

UPDATE STATISTICS *имя_схемы.имя_таблицы*

Вариант 08.

Создать хранимую процедуру без параметров, которая осуществляет поиск потенциально опасных ключевых слов в хранимых процедурах в текущей базе данных. В данном случае таким ключевым словом является 'EXEC'. Хранимая процедура выводит инструкцию 'EXEC', которая выполняет командную строку — строку символов или один из следующих модулей: хранимую процедуру или скалярную пользовательскую функцию. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию об объектах базы данных можно извлечь из системного представления sys.objects.
2. Информацию о модулях базы данных можно извлечь из системного представления sys.sql_modules.
3. Для работы со строками воспользоваться следующими функциями:

CHARINDEX (expression1 ,expression2 [, start_location])
Выполняет поиск выражения expression1 в выражении expression2 и в случае обнаружения возвращает его начальную позицию. Поиск начинается с аргумента start_location.

SUBSTRING (value_expression , start_expression , length_expression)
Возвращает фрагмент символьного, двоичного, текстового или графического выражения.

DATALENGTH (expression)
Возвращает число байтов, использованных для представления выражения.

Вариант 09.

Создать хранимую процедуру с выходным параметром, которая уничтожает все SQL DDL триггеры (триггеры типа 'TR') в текущей базе данных. Выходной параметр возвращает количество уничтоженных триггеров. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Для уничтожения триггера воспользоваться инструкцией DROP TRIGGER.
2. Системное представление sys.triggers содержит по одной строке для каждого объекта, являющегося триггером типа TR или TA. Имена триггеров DML существуют в пределах схемы и, следовательно, видимы в представлении sys.objects. Область существования имен триггеров DDL определяется родительской сущностью, поэтому эти имена видимы только в этом представлении. Столбцы parent_class и name однозначно идентифицируют триггер в базе данных.

Вариант 10.

Создать хранимую процедуру с выходным параметром, которая уничтожает все SQL DML триггеры (триггеры типа 'TR') в текущей базе данных. Выходной параметр возвращает количество уничтоженных триггеров. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Для уничтожения триггера воспользоваться инструкцией DROP TRIGGER.
2. Системное представление sys.triggers содержит по одной строке для каждого объекта, являющегося триггером типа TR или TA. Имена триггеров DML существуют в пределах схемы и, следовательно, видимы в представлении sys.objects. Область существования имен триггеров DDL определяется родительской сущностью, поэтому эти имена видимы только в этом представлении. Столбцы parent_class и name однозначно идентифицируют триггер в базе данных.

Вариант 11.

Создать хранимую процедуру с выходным параметром, которая уничтожает все SQL функции пользователя (функции типа 'FN', 'IF' и 'TF') в схеме 'dbo' в текущей базе данных. Выходной параметр возвращает количество уничтоженных функций. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Для удаления функций воспользоваться инструкцией DROP FUNCTION.
2. Информацию о пользовательских функциях можно извлечь из системного представления sys.objects.

Вариант 12.

Создать хранимую процедуру с выходным параметром, которая выводит текст на языке SQL всех скалярных SQL функций пользователя (функции типа 'FN') в текущей базе данных, имена которых начинаются с префикса 'ufn'. Выходной параметр возвращает количество найденных функций. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию об объектах базы данных можно извлечь из системного представления sys.objects.
2. Информацию о модулях базы данных можно извлечь из системного представления sys.sql_modules.

Вариант 13.

Создать хранимую процедуру с выходным параметром, которая выводит список имен и параметров всех скалярных SQL функций пользователя (функции типа 'FN') в текущей базе данных. Имена функций без параметров не выводить. Имена и список параметров должны выводиться в одну строку. Выходной параметр возвращает количество найденных функций. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию об объектах базы данных можно извлечь из системного представления sys.objects.
2. Информацию о параметрах объектов можно извлечь из системного представления sys.parameters.

Вариант 14.

Создать хранимую процедуру с выходным параметром, которая уничтожает все представления в текущей базе данных, которые не были зашифрованы. Выходной параметр возвращает количество уничтоженных представлений. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Для уничтожения представления воспользоваться инструкцией DROP VIEW.
2. Информацию об объектах базы данных можно извлечь из системного представления sys.objects.
3. Информацию о модулях базы данных можно извлечь из системного представления sys.sql_modules.

Вариант 15.

Создать хранимую процедуру с параметром, в которой для экземпляра SQL Server создаются резервные копии всех пользовательских баз данных, созданных или переименованных после даты, указанной в параметре процедуры. Имя файла резервной копии должно состоять из имени базы данных и даты создания резервной копии, разделенных символом нижнего подчеркивания. Дата создания резервной копии должна быть представлена в формате YYYYDDMM. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Для создания резервной копии базы данных воспользоваться инструкцией BACKUP DATABASE, имеющей следующий формат:

BACKUP DATABASE *имя_базы_данных* TO DISK = *полная_спецификация_файла_резервной_копии*

2. Имена баз данных экземпляра SQL Server можно извлечь из системного представления sys.databases.

Вариант 16.

Создать хранимую процедуру с входным параметром – имя таблицы, которая выводит сведения об индексах указанной таблицы в текущей базе данных. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Сведения об индексах указанной таблицы в указанной базе данных можно получить с помощью динамической административной функции sys.dm_db_index_physical_stats, принимающей пять параметров:
    1. Идентификатор базы данных - можно получить с помощью функции метаданных DB_ID.
    2. Идентификатор таблицы - можно получить с помощью функции метаданных OBJECT_ID.
    3. Для прочих параметров можно указать значения NULL.
2. Если имя таблицы задано неверно, напечатать сообщение 'Invalid table'.

Вариант 17.

Создать хранимую процедуру с входным параметром, которая выводит имена хранимых процедур, созданных с параметром WITH RECOMPILE, в тексте которых на языке SQL встречается строка, задаваемая параметром процедуры. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию об объектах базы данных можно извлечь из системного представления sys.objects.
2. Информацию о модулях базы данных можно извлечь из системного представления sys.sql_modules.

Вариант 18.

Создать хранимую процедуру без параметров, которая перестраивает все индексы всех таблиц в текущей базе данных. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию о таблицах базы данных можно извлечь из системного представления sys.objects.
2. Для перестроения индексов для таблицы в указанной базе данных воспользоваться командой DBCC DBREINDEX.
3. Использовать значение коэффициента заполнения для индекса, указанное при создании индекса.

Вариант 19.

Создать хранимую процедуру без параметров, которая осуществляет поиск ключевого слова 'EXEC' в тексте хранимых процедур в текущей базе данных. Хранимая процедура выводит инструкцию 'EXEC', которая выполняет хранимую процедуру или скалярную пользовательскую функцию. Созданную хранимую процедуру протестировать.

Рекомендации.

1. Информацию об объектах базы данных можно извлечь из системного представления sys.objects.

2. Информацию о модулях базы данных можно извлечь из системного представления sys.sql_modules.
3. Для работы со строками воспользоваться следующими функциями:

CHARINDEX (expression1 ,expression2 [, start_location])
Выполняет поиск выражения expression1 в выражении expression2 и в случае обнаружения возвращает его начальную позицию. Поиск начинается с аргумента start_location.

SUBSTRING (value_expression , start_expression , length_expression)
Возвращает фрагмент символьного, двоичного, текстового или графического выражения.

DATALENGTH (expression)
Возвращает число байтов, использованных для представления выражения.

Вариант 20.

Создать хранимую процедуру с входным параметром – имя базы данных, которая выводит имена ограничений CHECK и выражения SQL, которыми определяются эти ограничения CHECK, в тексте которых на языке SQL встречается предикат 'LIKE'. Созданную хранимую процедуру протестировать.

Рекомендации.
3. Информацию об объектах базы данных можно извлечь из системного представления sys.objects.
4. Информацию об ограничениях CHECK можно извлечь из системного представления sys.check_constraints.
5. Если имя базы данных задано неверно, напечатать сообщение 'Invalid database'.