|  |  |
| --- | --- |
| **1. Что такое первичные файлы? Какое у них расширение?**  Первичный файл содержит сведения, необходимые для запуска бд и ссылки на другие файлы в бд. В каждой бд имеется 1 первичный файл(он является обязательным), находящийся в первичной файловой группе(on primary). Расширение MDF. | **2.Что такое *вторичные файлы*? Какое у них расширение?**  Вторичные файлы не являются обязательными. Это пользовательские файлы, в которых хранятся данные пользователя. Расширение NDF. |
| **3.Что такое файловые группы? В каких случаях они используются?**  Файловые группы – поименованный набор файлов бд. Используются в основном для упрощения администрирования бд. Существуют операторы, позволяющие рассматривать файловую группу как единое целое и выполнять операции не с отдельными файлами, а сразу для файловой группы. | **4.Что значит *файловая группа* *по умолчанию*?**  В эту группу назначаются объекты, созданные без указания файловой группы, к которой они относятся. Файловая группа PRIMARY является группой по умолчанию, если она не была изменена инструкцией ALTER DATABASE. |
| 5.**Объясните назначение *временных таблиц* и принцип работы с ними. Поясните разницу между *локальными* и *глобальными* временными таблицами.**  Временные таблицы отличаются от постоянных тем, что они хранятся в системной бд TEMPDB и не могут иметь внешние ключи. Создаются для временного хранения результатов SELECT-запросов. Локальные отличают от глобальных форматом имени, областью видимости и жизненным циклом. Локальные начинаются с # и доступны создавшему их пользователю и применяются обычно для трудоемких select-запросов. Глобальные начинаются с ##, доступны всем пользователям, подключенным к серверу, применяются для обмена данными между несколькими сеансами. | **6.Что такое *пакет*?**  Пакет – группа операторов T-SQL, завершающаяся оператором GO. Пакет – это группа операторов T-SQL, которая обрабатывается сервером СУБД вместе. Понятие пакета необходимо при клиент-серверной архитектуре приложения. В этом смысле, пакет – это группа операторов T-SQL, которую программа клиент отправляет серверу на исполнение. |
| **7.Каким образом *объявляются переменные* в T-SQL? Какие *правила именования переменных* T-SQL?**  Для объявления переменных предназначен оператор DECLARE. Для каждой переменной указывается имя и тип. Имя должно начинаться с @. Нельзя с @@, т. к. зарезервировано для системных переменных. Жизненный цикл переменных завершается с завершением пакета. | **8.Каким образом можно *присвоить значения переменным* в T-SQL?**  Переменной, объявленной в операторе DELCLARE, можно присвоить значение тремя способами: инициализировать в операторе DECLARE, присвоить значение с помощью операторов SET или SELECT. |
| **9.Каким образом можно *вывести и распечатать значения переменных* в результирующий набор?**  Вывод возможен с помощью SELECT(можно формировать выходной результирующий набор) и с помощью PRINT(можно вывести строку в стандартный выходной поток). Каждый select формирует отдельный результирующий набор, а print – строку в единственный выходной поток. | **10.Поясните смысл глобальных переменных** @@ROWCOUNT – число обработанных строк  @@ERROR – код последней ошибки  @@SERVERNAME – имя локального сервера  @@IDENTITY – значение идентификатора, вставленного последним  @@SPID – идентификатор сеанса для текущего пользовательского процесса  @@VERSION – версия SQL Server  @@TRANCOUNT – уровень вложенности транзакций  @@FETCH\_STATUS – проверка результата считывания строк результирующего набора  @@NESTLEVEL – уровень вложенности текущей процедуры |
| **11.Объясните назначение функций**  *ERROR\_NUMBER – код последней ошибки*  *ERROR\_MESSAGE – сообщение об ошибке*  *ERROR\_LINE –номер строки с ошибкой(код последней ошибки)*  *ERROR\_PROCEDURE – имя процедуры или NULL*  *ERROR\_SEVERITY – уровень серьезности ошибки*  *ERROR\_STATE* – метка ошибки | **12.Каким образом можно выполнять *преобразование типов* данных в T-SQL?**  Существую 2 функции: CAST и CONVERT. Пости не отличаются. Преобразование явное и неявное. При явном указывается необходимый тип данных (cast(выражение as тип)) |
| **13.Что такое *блок операторов* T-SQL? Как он оформляется в программе?**  Блок позволяет создавать программную единицу из одного или нескольких операторов. Каждый блок начинается с begin и заканчивается оператором end. Блок может быть использован внутри if, тогда появляется возможность выполнения более одного оператора в зависимости от некоторого условия. | **14.Каким образом выполняется *конкатенация строк* в T-SQL?**  Конкатенация (объединение) строк происходит с использованием символа +.или +=. |
| **15.Назовите *операторы сравнения* (управления)T-SQL, поясните возможные конструкции этих операторов.**  =равно  >больше чем <меньше чем >=больше или равно <=меньше или равно <>не равно !=не равно  !<не меньше !>не больше  Результат их выполнения имеет тип Boolean(true,false,unknown)  Операторы можно применять ко всем выражениям, кроме выражений с типами text,ntext,image. Сравнение с null допустимо только с использованием is(not)null. Сравнение чисел и денежных естественным образом для чисел. Врмя и дата больше те, которые стоят правее в хронологической оси, строки: сравниваются 2 числа, образованные кодами символов. | 16. **Объясните назначение и синтаксис оператора выбора *CASE*.**  Case обычно применяется в select и служит для формирования одного из нескольких возможных значений. У case 2 формата: сравнивает результат одного выражения со списком заданных значений или вычисляет набор логических выражений.  Case вычисляемое\_выражение  When значение\_для\_сравнения then результирующ\_выражение  [else результирующее\_выражение]  end |
| **17.Каким образом можно организовать *цикл* в программе на T-SQL?**  While предназначен для организации программного цикла. Содержит 2 составляющие: логическое выражение и тело цикла. Логическое выражение задает условие выполнения тела цикла, а тело содержит 1 или более операторов, которые выполняются, пока логическое выражение возвращает true. Если в теле более 1 оператора, необходимо обернуть их в begin..end. Continue прерывает выполнение блока в теле и передает управление в начало цикла. Break прекращает работу цикла и передает управление оператору за while. | **18.Поясните принцип работы оператора *WAITFOR*.**  С помощью waitfor можно приостановить выполнение пакета. Работает в 2 режимах: Delay и Time. Delay приостанавливает выполнение пакета на заданный интервал времени, а Time приостанавливает выполнение до наступления заданного значения системного времени на сервере. |
| **19.Объясните назначение и способ применения конструкции *TRY/CATCH*.**  Для обработки ошибок существует конструкция из 2 блоков: Try и Catch. Try содержит код, в котором могут возникнуть ошибки, а в Catch – код для обработки ошибок. В Catch можно использовать несколько системных функций для диагностики ошибки.  Begin try  Инструкции t-sql  End try  Begin catch  [инструкции t-sql]  End catch | **20.Дайте определение *курсору*. Какие *типы курсоров* бывают?**  Курсор – механизм обработки результирующего набора select-запроса. Курсор – поименованная область памяти, содержащая результирующий набор select-запроса. Курсоры бывают локальными и глобальными, статическими и динамическими. |
| **21.Объясните *схему работы с курсором*.**  1.объявление курсора в declare  2.открытие курсора с помощью open  3.с помощью fetch считывается 1 или несколько строк результирующего набора, связанного с курсором select-оператора. Результат каждого считывания проверяется с помощью системной функции @@FETCH\_STATUS  4.Закрытие курсора с помощью close  5.Если курсор глобальный, то он должен быть освобожден с помощью DEALLOCATE. | **22.Объясните отличия между *GLOBAL* и *LOCAL*-курсорами.**  Локальный курсор может применяться в рамках одного пакета и ресурсы, выделяемые ему при объявлении освобождаются сразу после завершения работы пакета. Признак локального курсора – атрибут LOCAL, указываемый при объявлении.  Глобальный может быть объявлен, открыт, использован в разных пакетах. Выделенные ему ресурсы освобождаются только после выполнения оператора DEALLOCATE или при завершении сеанса пользователя. Для объявления курсора глобальным используется атрибут GLOBAL. По умолчанию курсор глобальный, но значение это должно быть установлено с помощью ALTER DATABASE SET CURSOR\_DEFAULT. |
| **23.Объясните отличия между *STATIC* и *DYNAMIC*-курсорами.**  При объявлении статического курсора должен указываться атрибут STATIC. Открытие статического курсора приводит к выгрузке результирующего набора строк в динамически созданную временную таблицу системной бд TEMPDB и все дальнейшие курсорные операции осуществляются с этой таблицей. Изменения в исходных таблицах не буду отражаться в курсорном результирующем наборе. Если курсор динамический, то изменения в исходных таблицах отражаются в результирующем наборе. @@CURSOR\_ROWS после открытия динамического курсора всегда -1. | **24.Объясните назначение встроенной функции *@@CURSOR\_ROWS*.**  С помощью этой функции можно получить количество строк в результирующем наборе статического курсора. После открытия динамического курсора эта функция возвращает -1.  -m - курсор заполнен асинхронно. –m=количеству строк, находящихся в наборе ключей  0 - ни один курсор не был открыт, не было строк для последнего открытого курсора  n – курсор полностью заполнен. n=общему количеству строк в курсоре |
| **25.Объясните назначение встроенной функции *@@FETCH\_STATUS*.**  Возвращает состояние последней инструкции FETCH, вызванной в любом курсоре, открытом в данном соединении.  0 – оператор FETCH выполнен успешно  -1 – достигнут конец результирующего набора. Строка не считывается  -2 – выбранная строка отсутствует в бд. | **26.Объясните назначение атрибута курсора *READ ONLY*.**  Свойство READ\_ONLY запрещает применение операции CURRENT OF в секции WHERE операторов DELETE и UPDATE. Однако свойство READ\_ONLY не мешает применять эти операторы без использования CURRENT OF. Атрибут READ\_ONLY имеет смысл только для динамических и ключевых курсоров, статические курсоры по определению применяются только для чтения. |
| **27.Объясните назначение атрибута курсора *SCROLL*.**  По умолчанию для курсора установлен атрибут SCROLL, позволяющий применять оператор FETCH с дополнительными опциями позиционирования. Указывает, что доступны все параметры выборки (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE). | **28.Объясните назначение следующих операторов *FETCH* *LAST*/*FIRST*/*ABSOLUTE*/*RELATIVE*/*NEXT*/*PRIOR***  **NEXT** Возвращает строку результата сразу же за текущей строкой и перемещает указатель текущей строки на возвращенную строку.  NEXT является параметром по умолчанию выборки из курсора.**PRIOR** Возвращает строку результата, находящуюся непосредственно перед текущей строкой и перемещает указатель текущей строки на возвращенную строку. Если инструкция FETCH PRIOR выполняет первую выборку из курсора, не возвращается никакая строка и положение курсора остается перед первой строкой.**FIRS**T Возвращает первую строку в курсоре и делает ее текущей.**LAST** Возвращает последнюю строку в курсоре, и делает ее текущей.**ABSOLUTE { *n*| @*nvar*}** Если *n* или @*nvar* имеет положительное значение, то возвращается строка, отстоящая на *n* строк от начала курсора, а возвращенная строка становится новой текущей строкой. Если *n* или @*nvar* имеет отрицательное значение, то возвращается строка, отстоящая на *n*строк от конца курсора, а возвращенная строка становится новой текущей строкой. Если *n* или @*nvar* равно 0, не возвращается ни одна строка. Аргумент *n* должен быть целочисленной константой, а @*nvar* — иметь тип **smallint**, **tinyint** или **int**.  **RELATIVE { *n*| @*nvar*}** Если *n* или @*nvar* имеет положительное значение, то возвращается строка, отстоящая на *n* строк от текущей строки, а возвращенная строка становится новой текущей строкой. Если *n* или @*nvar* имеют отрицательное значение, то возвращается строка, предшествующая на *n* строк текущей строке, а возвращенная строка становится новой текущей строкой. Если *n* или @*nvar* равно 0, то возвращается текущая строка.  Аргумент *n* должен быть целочисленной константой, а @*nvar* — иметь тип **smallint**, **tinyint** или **int**. |
| **29.Объясните назначение и принцип работы конструкций UPDATE/DELETE *WHERE CURRENT OF*.**  Для курсоров с установленным свойством FOR UPDATE, помимо чтения данных из строк с помощью оператора FETCH, можно эти строки изменять или удалять с помощью специального формата операторов UPDATE и DELETE.  В секции WHERE операторы DELETE и UPDATE используют операцию CURRENT OF, для которой указывается имя курсора. Такой формат операторов позволяет удалять или изменять строки в таблице, соответствующие текущей позиции курсора в результирующем наборе. | **30.Что такое *транзакция*? В каких случаях механизм транзакции следует применять?**  **Транзакция** − это механизм базы данных, позволяющий таким образом объединять несколько операторов, изменяющих базу данных, чтобы при выполнении этой совокупности операторов они или все выполнились или все не выполнились. Устанавливает согласованность бд. |
| **31.Объясните понятие *ACID*.**  Обычно при определении понятия транзакции используют перечень требований к этому механизму, сформулированный в конце 70-х гг. Джимом Греем в виде аббревиатуры ACID: Atomicity – атомарность, Consistency – согласованность, Isolation – изолированность, Durability – долговечность. | **32.Перечислите *все операторы*, которые используются для управления транзакцией. К какой *группе* операторов SQL они относятся?**  Транзакция – это несколько DML-операторов, которые либо все успешно выполнятся, либо все не выполняются. TCL SQL включает четыре оператора: BEGIN TRAN, SAVE TRAN, COMMIT TRAN и ROLLBACK TRAN |
| **33.Поясните понятия *фиксация транзакции* и *откат транзакции*.**  Оператор COMMIT TRAN фиксирует транзакцию – подтверждает ее успешное завершение. Оператор ROLLBACK TRAN откатывает транзакцию – отменяет все изменения, произведенные DML-операторами в таблицах БД в рамках данной транзакции. | **34.Поясните понятие *вложенная транзакция*.**  Транзакция, выполняющаяся в рамках другой транзакции, называется вложенной. При работе с вложенными транзакциями следует учитывать следующие свойства.   1. COMMIT вложенной транзакции действует только на внутренние операции вложенной транзакции. 2. ROLLBACK внешней транзакции отменяет зафиксированные операции внутренней транзакции. 3. ROLLBACK вложенной транзакции действует на операции внешней и внутренней транзакции, а также завершает обе транзакции. 4. Уровень вложенности транзакции можно определить с помощью встроенной системной функции @@TRANCOUT. |
| **35.Что такое *контрольная точка транзакции*.**  Если транзакция состоит из нескольких относительно независимых блоков операторов T-SQL, изменяющих БД, то может быть использован оператор SAVE TRANSACTION, формирующий контрольную точку транзакции. Контрольная точка – это метка в явной транзакции, определяющая момент времени, к которому может быть выполнен откат. | **36.С помощью какого *оператора* устанавливается *уровень изолированности* *транзакций*?**  Для установки уровня изолированности текущей для транзакций в текущем подключении служит инструкция SET TRANSACTION ISOLATION LEVEL. |
| **37.Перечислите *все уровни изолированности транзакций* и поясните их действие.**  Пять уровней изолированности транзакций: READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SNAPSHOT и SERIALIZABLE. Для установки уровня изолированности текущей для транзакций в текущем подключении служит инструкция SET TRANSACTION ISOLATION LEVEL.  уровень READ COMMITTED (действует по умолчанию) не допускает неподтвержденного чтения, но при этом возможно неповторяющееся и фантомное чтение, а уровень SERIALIZABLE решает все три проблемы.  Свойства уровней изоляции транзакций   |  |  |  |  | | --- | --- | --- | --- | | Уровень изоляции | Непд  твеж  деное  чтение | Непов  Торяющееся чтение | Фантомное чтение | | READ UNCOMMTITTED | + | + | + | | READ COMMITED | – | + | + | | REPEATABLE READ | – | – | + | | SNAPSHOT | – | – | – | | SERIALIZABLE | – | – | – | |  |  |  |  | | **38.Поясните назначение функции *@@TRANCOUNT*.**  @@TRANCOUNT возвращает уровень вложенности транзакции. Значение, превышающее нуль, означает, что транзакция не завершена. |
| **39.Что такое *индекс*? Для чего индексы используются в базах данных?**  Индекс – это объект БД, позволяющий ускорить поиск в определенной таблице. Предназначены для ускорения выполнения запросов. | **40.В каких случаях *индексы* создаются *неявно* (автоматически)?**  По умолчанию для таблицы БД со столбцом, имеющим свойство PRIMARY KEY (ограничение целостности – первичный ключ), при создании таблицы автоматически создается кластеризованный уникальный индекс.  Если при этом в свойствах столбца первичного ключа специально не указано ключевое слово NONCLUSTERED, то автоматически создастся кластеризованный индекс. |
| **41.Чем отличается *кластеризованный* индекс от *некластеризованного*?**  Некластеризованные индексы в отличие от кластеризованных никак не влияют на физический порядок строк в таблице. Как и в случае кластеризованных индексов, данные индекса организованы в виде сбалансированного дерева. С таблицей некластеризованный индекс связан указателями на ее строки. Для одной таблицы БД допускается создавать до 1023 некластеризованных индексов. | **42.Поясните понятия *план запроса* и *стоимость запроса* SQL.**  План запроса представляет собой алгоритм выполнения SQL-запроса. На каждом шаге алгоритма выполняется элементарное действие сервера СУБД. При построении плана запроса для каждого шага вычисляется числовая величина, пропорциональная продолжительности выполнения шага, называемая стоимостью. Суммарная стоимость шагов плана составляет стоимость всего запроса и, соответственно, является величиной, пропорциональной продолжительности выполнения запроса. |
| **43.Что такое *хранимая процедура*? Поясните понятия *входной параметр* и *возвращаемый* (выходной параметр) процедуры.**  Хранимая процедура – это объект БД, представляющий собой поименованный код T-SQL. Хранимая процедура может использовать не только входные параметры, через которые передаются значения в процедуру, но и выходные параметры(ключевое слово OUTPUT). Выходные параметры по сути представляют результат работы процедуры, те значения, которые она возвращает пользователю. | **44.Что такое *функция*? *скалярная* *функция*? *табличная* *функция*?**  Функция – это объект БД, представляющий собой поименованный код T-SQL. Скалярная функция возвращает единственное значение и принимает любое количество параметров. Допускается возвращать и принимать все типы данных, за исключением TIMESTAMP и устаревших IMAGE, TEXT и NTEXT. Встроенные табличные функции возвращают к точке вызова результирующий набор SELECT-запроса и по своим возможностям очень напоминают представления. Многооператорные табличные функции возвращают к точке вызова результирующий набор и допускают более сложную логику работы. |
| **45.Что такое *представление*?**  Представление (от англ. View) – это объект БД, представляющий собой поименованный SELECT-запрос. | **46.Поясните назначение опции *WITH CHECK OPTION*.**  CHECK OPTION Обеспечивает соответствие всех выполняемых для представления инструкций модификации данных критериям, заданным при помощи аргумента select\_statement. Если строка изменяется посредством представления, предложение WITH CHECK OPTION гарантирует, что после фиксации изменений доступ к данным из представления сохранится.  Чтобы операция вставки не могла осуществиться в том случае, когда информация не удовлетворяет условию, записанному в секции Where, то следует создавать представление с опцией WITH CHECK OPTION. |
| **47.Что такое *триггер*?**  Триггер – это особый вид хранимой процедуры, предназначенной для обработки событий в БД. | **48.Что такое *DLL*(системный)-триггер?**  Различают два вида DDL-триггеров: триггеры уровня сервера (ALLSERVER), триггеры уровня БД (DATABASE). Триггеры уровня сервера могут обрабатывать события сервера СУБД (создание, изменение или удаление объектов сервера, подключение к серверу и пр.). Триггеры уровня базы данных предназначены для обработки событий, происходящих в рамках БД. События DDL-триггеров являются иерархическими. |
| **49.Что такое *DML*-триггер?**  Каждый DML-триггер связан с таблицей или представлением и предназначен для обработки одного или нескольких событий, соответствующих трем операторам, изменяющим содержимое таблицы: INSERT, UPDATE и DELETE. MSS поддерживает два типа DML-триггеров: AFTER и INSTEAD OF. | **50.Что такое AFTER-триггер?**  AFTER-триггеры. Триггеры типа AFTER исполняются после выполнения оператора, вызвавшего соответствующее событие. При создании триггера указывается: 1) имя таблицы (после ключевого слова ON), с которой этот триггер связан; 2) событие, обработку которого осуществляет триггер. |
| **51.Что такое *INSTEAD* OF-триггер.**  INSTEAD OF-триггеры. Триггеры типа INSTEAD OF исполняются вместо оператора, вызвавшего соответствующее событие. Следует отметить, для таблицы или представления допускается создание только по одному INSTEAD OF-триггеру, реагирующему на каждое событие. Кроме того, выполнение INSTEAD OF-триггера предшествует проверке установленных для таблицы ограничений целостности | **52.Что такое *XML*? Для чего используется XML?**  XML(ExtensibleMarkupLanguage)расширяемый язык разметки. Язык XML задает общие правила, по которым создаются теги и оформляются XML-документы. XML-документ представляет собой текстовый файл, содержащий данные, разделенные тегами, составленными по правилам XML и расположенными в правильном порядке. |
| **53.Что значит: правильно построенный XML-документ? Опишите основные синтаксические правила правильно построенного XML-документа.**  Правильно построенный XML-документ – т. е. в соответствии с правилами XML. Правила:  1) XML-документ начинается специальным тегом с именем ?xml, содержащим информацию о версии XML и используемой кодировке;  2) каждый элемент начинается открывающим тегом и завершается закрывающим тегом;  3) между открывающим и закрывающим тегами некоторых элементов могут находиться другие теги;  4) XML-документ является содержимым единственного элемента, называемого корневым;  5) применение XML-формата влечет к значительному увеличению избыточности данных;  6) XML-формат позволяет создавать самодокументированные данные. | **54.В каких *режимах* осуществляется *преобразование* результата SELECT-запроса в формат XML? В чем их отличия?**  Для преобразования результата SELECT-запроса в формат XML в операторе SELECT применяется секция FOR XML. При этом можно использовать один из четырех режимов: RAW, AUTO, PATH и EXPLICIT.По умолчанию в режиме RAW в результате SELECT-запроса создается XML-фрагмент, состоящий из последовательности элементов с именем row. Каждый элемент row соответствует строке результирующего набора, имена его атрибутов совпадают с именами столбцов результирующего набора (из списка SELECT), а значения атрибутов равны их значениям.Результат, полученный в режиме AUTO для простых SELECT-запросов, похож на результат, полученный в режиме RAW. Основное отличие – в качестве имени элемента, соответствующего строке исходной таблицы, используется ее имя. Особенность режима AUTO проявляется в многотабличных запросах. В этом случае режим AUTO позволяет построить XML-фрагмент с применением вложенных элементов. Причем порядок вложенности основывается на порядке столбцов (слева направо), указанных в SELECT-списке.  Режим PATH позволяет разработчику наиболее полным образом управлять процессом формирования XML-структуры. Каждый столбец конфигурируется независимо с помощью заданного в формате XPATH имени псевдонима этого столбца.Режим EXPLICIT преобразует набор строк, получаемый в результате выполнения запроса, в XML-документ. Для того чтобы режим EXPLICIT создал XML-документ, набор строк должен иметь определенный формат. То есть необходимо написать запрос SELECT для создания набора строк, универсальной таблицы, имеющей определенный формат, так чтобы логика обработки могла создать желаемый XML. |
| **55.Перечислите *формы преставления* данных в XML-документа? В чем их разница?** | **56.Назовите функцию, с помощью которой можно сформировать строки в реляционной таблице из *XML*-документа?**  функция OPENXML. Она принимает три входных параметра: дескриптор, сформированный системной хранимой процедурой с именем SP\_XML\_PREPAREDOCUMENT; выражение XPATH и целое положительное число, определяющее режим работы функции. |
| **57.Для чего используются *XML*-столбцы в реляционной таблице? Какие существуют ограничения на применение *XML*-столбцов?**  XML подходит, если нужна не зависящая от платформы модель, гарантирующая совместимость данных за счет применения структурной и семантической разметки. Уместен, если:   * данные разрежены, их структура неизвестна или их структура может значительно измениться в будущем; * данные представляют иерархию контейнеров, а не ссылки между сущностями, и могут быть рекурсивными; * данные упорядочены; * требуется запрашивать данные или обновлять их фрагменты на основе их структуры.   Если ни одно из этих условий не выполняется, следует выбрать реляционную модель данных. Хранение данных в XML-столбце обеспечивает проверку структуры и правильности данных, поддержку детализированных запросов и обновлений XML-данных. Ограничения типа данных xml   * Размер хранимого представления экземпляра типа данных xml не должен превышать 2 ГБ. * Данный тип не может быть использован в качестве подтипа экземпляра sql\_variant . * Приведение или преобразование к типам text или ntext не поддерживается. Вместо этого используются типы varchar(max) илиnvarchar(max). * Сравнение и сортировка не допускаются. Это значит, что данные типа xml не могут использоваться в предложении GROUP BY. * Не может использоваться в качестве параметров любых скалярных встроенных функций, кроме ISNULL, COALESCE и DATALENGTH.   Не может использоваться как ключевой столбец индекса. Однако может включаться в виде данных в кластеризованный индекс или явно добавляться в некластеризованный индекс при его создании с помощью ключевого слова INCLUDE. | **58.Для чего используется XML-Schema?**  XML-Schema – это одна из реализаций языка XML, поддерживаемая консорциумом W3C и предназначенная для описания структуры XML-документа. С помощью языка XML-Schema можно описать правила, которым должен подчиниться XML-документ. Файл, содержащий XML-Schema, обычно имеет расширение XSD. |
| **59. Что означает: типизированный XML-столбец?**  В типизированных XML-столбцах, параметрах и переменных можно хранить XML-документы или содержимое. Во время объявления необходимо указать при помощи флага, что хранится: документ или содержимое. Кроме того, необходимо предоставить системе коллекцию XML-схем. Укажите флаг DOCUMENT, если каждый экземпляр XML имеет ровно один элемент верхнего уровня. В противном случае укажите флаг CONTENT. Компилятор запросов использует флаг DOCUMENT при проверке типов во время компиляции запросов для определения одинарных элементов верхнего уровня. | **60. Для чего применяются типизированные XML-столбцы?**  В следующих ситуациях следует использовать типизированный тип данных xml :   * есть схемы XML-данных и требуется, чтобы сервер проверял соответствие данных этим схемам; * требуется оптимизировать хранение данных и обработку запросов на основе информации о типах;   требуется в более полной мере использовать информацию о типах при компиляции запросов. |