* Индексы бывают кластеризованные, некластеризованные, уникальные, неуникальные,
* Индексы нужны для ускорения запросов, так как СУБД не придется перебирать все данные в таблице, а искать по В+ дереву, что значительно облегчает задачу(почитать)
* кластеризованные – листья дерева индекса содержат страницы данных страницы
* Некластеризованные –листьев состоят из ключей индекса и закладок
* некластеризованные составные, некластеризованные покрытия, некластеризованные фильтруемые,
* Процесс образования неиспользуемых фрагментов памяти называется *фрагментацией*.
* Реоргагизация, перестройка
* Параметр FILLFACTOR указывает процент заполнения индексных страниц нижнего уровня
* **Курсор** является программной конструкцией, которая дает возможность пользователю обрабатывать строки результирующего набора запись за записью. Курсоры бывают *локальные* и *глобальные* (по умолчанию), *статические* и *динамические* (по умолчанию).
* *Локальный* курсор может применяться в рамках одного пакета и ресурсы, выделенные ему при объявлении, освобождаются сразу после завершения работы пакета.
* *Глобальный* курсор может быть объявлен, открыт и использован в разных пакетах. Выделенные ему при объявлении ресурсы освобождаются только после выполнения оператора DEALLOCATE или при завершении сеанса пользователя
* Открытие *статического* курсора приводит к выгрузке результирующего набора во временную таблицу системной БД **TEMPDB**, и все дальнейшие операции осуществляются с этой таблицей.
* Если заменить LOCAL STATICна LOCAL DYNAMIC, то изменения будут отражаться в результирующем наборе.
* Функция @@CURSOR\_ROWS сообщает число строк в курсоре. @@CURSOR\_ROWS

-n – количество записей при асинхронной выборке,

n – количество записей при синхронной выборке,

0 – курсор не открыт

* RTRIM
* CURSOR\_STATUS – cкалярная функция, позволяющая при вызове хранимой процедуры определить, вернула ли она курсор и результирующий набор для данного параметра
* @@FETCH\_STATUS

0 – успешная выборка,

-1 – вышли за диапазон таблицы,

-2 – запись удалена после открытия курсора

* CURRENT OF позволяет удалять и обновлять запись которая была последней в курсоре.
* SCROLL позволяет перемещаться по индексам как с начала, так и с конца
* **Хранимая процедура** – это поименованный код на языке Transact-SQL. Хранимая процедура может быть создана с помощью CREATE, изменена с помощью ALTER и удалена с помощью оператора DROP. Процедура может принимать входные и формировать выходные параметры. Результатом ее выполнения может быть целочисленное значение, которое возвращается к точке вызова оператором RETURN, либо один или более результирующих наборов, сформированных операторами SELECT, либо содержимое стандартного выходного потока, полученного при выполнении операторов PRINT.
* Вызов процедуры осуществляется оператором EXECUTE (EXEC).
* Допускается применение:

Основных DDL, DML и TCL-операторов

Конструкций TRY/CATCH

Курсоров

Временных таблиц

* Не допускается применение:

USE databasename

Триггеры

Функции

Другие хранимые процедуры

Представление

* INSERT EXECUTE если нет выходных параметров, то можно использовать их для добавления в некоторую таблицу
* Кроме передачи результата выполнения через выходные параметры хранимая процедура также может возвращать какое-либо значение типа INT с помощью оператора RETURN. Хотя данная возможность во многом нивелирована использованием выходных параметров, через которые можно возвращать результат, тем не менее, если надо возвратить из процедуры одно значение, то вполне можно использовать оператор RETURN.
* **Функция** – это объект БД, представляющий собой поименованный код T-SQL. Для создания, удаления и изменения функций надо использовать операторы CREATE, DROP и ALTER соответственно
* В функции не допускается применение DDL-операторов, DML-операторов, изменяющих БД (INSERT, DELETE, UPDATE), конструкций TRY/CATCH, а также использование транзакций.
* Результатом выполнения функции является возвращаемое к точке вызова значение.
* Если функция возвращает единственное значение (число, строка, дата, время и пр.), то она называется *скалярной*.
* Функция, возвращающая таблицу, называется *табличной*.
* В зависимости от структуры кода, различают *встроенные* функции и *многооператорные* табличные функции.
* *многооператорные* табличные функции.????
* При вызове функции надо указывать ее имя с точностью до схемы БД. Если при создании функции имя схемы не указано, то она размещается по умолчанию в схеме **DBO**.
* **Триггер** – это особый вид хранимой процедуры, предназначенной для обработки событий в БД. Поддерживается два типа триггеров: *DDL-триггеры* и *DML-триггеры*.
* **DML-**триггеры бывают двух типов: AFTER-триггеры и INSTEAD OF-триггеры.
* Триггеры типа AFTER исполняются *после* выполнения оператора, вызвавшего соответствующее событие. При этом создаются автоматически две псевдотаблицы INSERTED и DELETED.
* Ж
* Ж
* Триггер типа INSTEAD OF выполняется *вместо* оператора, вызвавшего соответствующее событие. Выполнение INSTEAD OF триггера предшествует проверке установленных для таблицы ограничений целостности.
* Событие INSERT при выполнении AFTER-триггера приводит к тому, что в псевдотаблицу INSERTED помещаются строки, добавленные оператором INSERT, вызвавшим это событие. Псевдотаблица DELETED остается пустой.
* При возникновении события DELETE в таблицу DELETED копируются удаленные строки, а таблица INSERTED остается пустой.
* При изменении строк таблицы с помощью оператора UPDATE заполняются обе псевдотаблицы, при этом таблица INSERTED содержит обновленные версии строк, а таблица DELETED − версию строк до их изменения.
* **XML** (Extensible Markup Language) – расширяемый язык разметки.
* При работе с базами данных важными являются две задачи: преобразование *табличных данных в XML-структуры* и преобразование *XML-структур в строки реляционной таблицы*.
* Для преобразования результата SELECT-запроса в формат XML в операторе SELECT применяется секция FOR XML. При этом могут использоваться режимы RAW, AUTO, PATH.
* 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, а значения атрибутов равны их значениям.
* Особенность режима AUTO проявляется в многотабличных запросах. В этом случае режим AUTO позволяет построить XML-фрагмент с применением вложенных элементов.
* При использовании режима PATH каждый столбец конфигурируется независимо с помощью псевдонима этого столбца.
* функция **OPENXML**, которая принимает три входных параметра: дескриптор, выражение XPATH и целое положительное число, определяющее режим работы функции.
* Выражение XPATH предназначено для выбора требуемых данных из исходного XML-документа.

query() – выполнение запроса

exist() – проверка существования

value() – возвращает значение атрибута

nodes() – возвращает набор узлов

modify() – изменение документа

* Директивы: TYPE, ROOT, ELEMENTS
* сохранять результат реляционного запроса как XML-документ или фрагмент типа данных XML
* Добавление к результирующему набору XML одного элемента верхнего уровня
* **XML Schema** — язык описания структуры XML-документа – предназначен для определения правил, которым должен подчиняться документ

-----

* **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.
* Свойства транзакций:

[**Atomicity — Атомарность**](https://habr.com/ru/post/555920/#atomicity)

Атомарность гарантирует, что каждая транзакция будет выполнена полностью или не будет выполнена совсем. Не допускаются промежуточные состояния.

[**Consistency — Согласованность**](https://habr.com/ru/post/555920/#consistency)

Транзакция, достигающая своего нормального завершения (EOT — end of transaction, завершение транзакции) и, тем самым, фиксирующая свои результаты, сохраняет согласованность базы данных

[**Isolation — Изолированность**](https://habr.com/ru/post/555920/#isolation)

Во время выполнения транзакции параллельные транзакции не должны оказывать влияния на её результат.

[**Durability — Надёжность**](https://habr.com/ru/post/555920/#durability)

Если пользователь получил подтверждение от системы, что транзакция выполнена, он может быть уверен, что сделанные им изменения не будут отменены из-за какого-либо сбоя.

* Неявная транзакция начинается, если выполняется один из следующих операторов:
* CREATE, DROP; ALTER TABLE; INSERT, DELETE, UPDATE, SELECT, TRUNCATE TABLE; OPEN, FETCH; GRANT (выдача разрешений), REVOKE (запрещение разрешений).
* Неявная транзакция продолжается до тех пор, пока не будет выполнен оператор фиксации (COMMIT) или оператор отката (ROLLBACK) транзакции.
* Переключение в режим *явной транзакции* осуществляется с помощью оператора BEGIN TRANSACTION. Транзакцию должен завершать один из операторов: COMMIT TRAN или ROLLBACK TRAN. После завершения явной транзакции происходит возврат в исходный режим (автофиксации или неявной транзакции).
* Системная функция @@TRANCOUNT возвращает уровень вложенности транзакции. (если значение больше нуля, то транзакция не завершена).
* Изоляция решает 4 проблемы:

**Повторное чтение**

когда мы видим обновленные и удаленные строки (UPDATE, DELETE)

**Потерянная запись**

Первая транзакция, если не закомментирована, теряет свою свои свойства, если для те же данные параллельно изменяются в другой транзакции

**Фантомное чтение**

когда мы видим добавленные записи (INSERT).

**Грязное чтение**

Для этого уровня параллельно исполняющиеся транзакции видят только не зафиксированные изменения из других транзакций.

* Если транзакция состоит из нескольких независимых блоков операторов T-SQL, изменяющих базу данных, то может быть использован оператор SAVE TRANSACTION, формирующий *контрольную точку* транзакции.
* При работе с вложенными транзакциями нужно учитывать следующее:

− оператор COMMIT вложенной транзакции действует только на внутренние операции вложенной транзакции;

− оператор ROLLBACK внешней транзакции отменяет зафиксированные операции внутренней транзакции;

− оператор ROLLBACK вложенной транзакции действует на операции внешней и внутренней транзакции, а также завершает обе транзакции;