**CLR (Common Language Runtime - общеязыковая исполняющая среда)** — это основная часть .NET Framework, которая отвечает за выполнение программ, написанных на разных языках программирования, таких как C# и Visual Basic (VB). CLR обеспечивает управление исполнением этих программ, гарантируя их безопасность, производительность и совместимость.

В отличие от нативных программ, которые выполняются напрямую на операционной системе, программы, запускаемые через CLR, работают в управляемой среде. Это означает, что CLR автоматически управляет такими аспектами, как сборка мусора, типизация данных и безопасность выполнения, что снижает вероятность ошибок, связанных с памятью и ресурсами.

В SQL Server CLR используется для выполнения кода, написанного на языках .NET (например, C#). Это позволяет интегрировать более сложные логики и алгоритмы в базу данных, которые можно было бы сложно или невозможно реализовать с использованием только T-SQL.

Это механизмы, которые позволяют изолировать выполнение различных приложений внутри одного процесса. Домены приложений обеспечивают безопасность и стабильность, потому что приложения, работающие в разных доменах, не могут напрямую взаимодействовать друг с другом, что предотвращает распространение ошибок или утечек памяти.

Когда SQL Server выполняет запросы, возвращающие большие объемы данных, он может обрабатывать их построчно, по одной строке за раз, вместо того чтобы загружать весь результат в память. Это позволяет эффективно работать с большими объемами данных и уменьшает нагрузку на систему.

* Процедуры
* Функции – скалярные и табличные
* Типы
* Триггеры

**sp\_configure** — это системная хранимая процедура в SQL Server, которая позволяет управлять настройками конфигурации сервера.

**'clr enabled'** — это параметр конфигурации, который управляет возможностью использования SQL CLR в SQL Server.

**RECONFIGURE** — это команда, которая применяет все изменения, сделанные с помощью sp\_configure. Она фактически заставляет SQL Server принять новые настройки конфигурации.

В **SQL Server** есть встроенные агрегатные функции, такие как SUM, AVG, COUNT, которые используются для вычисления некоторых итоговых значений по набору данных. Например, чтобы подсчитать сумму всех значений в столбце, можно использовать стандартную функцию SUM.

Однако иногда требуется создать **свою собственную агрегацию**. Например, вы хотите подсчитать среднее значение, но с каким-то дополнительным расчетом, или создать сложный агрегат, который не поддерживается стандартными функциями. В таких случаях можно написать **пользовательскую агрегатную функцию (UDA)** с помощью **SQL CLR**.

* Init – инициализация корректными значениями при отсутствии строк
* Accumulate – вычисление агрегации, выполняется **для каждой строки** в наборе данных
* Merge – применение агрегата над агрегатом
* Terminate – возвращает конечный результат
* INullable
  + IsNull – проверка на NULL
  + Null – вновь созданный экземпляр типа
* ToString – строковое представление экземпляра типа
* Parse - получение экземпляра из строки
  + SAFE– не нужны внешние ресурсы, управляет SQL Server
  + EXTERNAL\_ACCESS – доступны внешние ресурсы (файлы, сетевые, службы, реестр)
  + UNSAFE – разрешены любые действия

**Сериализация** — это процесс преобразования объекта в формат, который может быть сохранен (например, в файл или передан по сети) и затем восстановлен обратно в объект.

В случае с SQL CLR, сериализация позволяет сохранять состояния объектов в базе данных, так как SQL Server не может напрямую работать с сложными типами данных, такими как структуры, классы или пользовательские типы данных. Чтобы хранить эти данные, их сначала нужно сериализовать в формат, который SQL Server может понять и эффективно хранить (например, в бинарном формате).

**IsDeterministic = true**:

* Указывает, что функция всегда возвращает один и тот же результат для одинаковых входных данных.

**IsPrecise = true**:

* Указывает, что функция возвращает точный результат. Например, если бы функция работала с типами данных с плавающей запятой, **IsPrecise = true** говорит SQL Server, что она не будет потерять точность при вычислениях.