**Конвейерная функция или pipelined-функция** – табличная функция, которая поставляет свои результаты клиенту по мере их подготовки. Не материализуют весь результирующий набор.

**Профит**

- можно сократить выполнение sql-запросов и pl/sql-кода, ЛИБО поставить результат гораздо раньше, чем это возможно с обычными функциями

- уменьшает затраты ресурсов при материализации результата (PGA)

(PGA может забиться одним лишь результирующим набором функций, когда, например, у нас коллекция состоит из объекта, а объект состоит из нескольких полей, причем строковые и длинные и все они заполнены + работают параллельные сессии)

**Синтаксис**

create or replace function func\_name(…) return тип\_коллекция

**pipelined**

is

begin

…цикл…

**pipe row** (элемент\_коллекции);

…

end;

От обычной функции отличается ключевой конструкцией pipelined.

Обязательно должна возвращаться коллекция

Её заполнение происходит по мере выполнения через pipe row (в pipe row указывается элемент коллекции)

Важный момент: результаты должны отдаваться итеративно в цикле. Это м.б., например, проход по курсору (так обычно бывает при обработке входных данных), либо просто цикл

Как только программа достигает участка с pipe row, результат будет передан в вызывающую среду. При этом вызывающая сторона уже может что-то делать с этой строкой.

**Пояснение к разнице выполнения простой и pipelined-функций из презентации (слайд 22-24)**

**\*не думаю, что это важно для экза, но если интересно рассмотреть разницу работы двух функций…**

**Пример 3.1**: простой вызов функции без фильтрации.

Вызываем простую функцию ------- отработал-о за 5 секунд

Вызываем конвейерную функцию ---------- отработало за 5 секунд

Обе функции выполняются за одно и то же время, им нужны все результаты. Нет никаких фильтров. Разница не очевидна

**Пример 3.2:** Вызов функции с ограничением количества строк через **rownum**

Ограничим выдачу запроса 2 строчками

Вызываем простую функцию ------- отработало за 5 секунд

Вызываем конвейерную функцию ---------- отработало за 2 секунды

Теперь разница более понятна

Т.к. конвейерная функция даёт результат по мере готовности строк, нет никакой необходимости отдавать все 5 строчек, чтобы получить результат запроса из двух строк. Просто не за чем дальше выполнять конвейерную функцию, если мы уже получили необходимый результат. В данном примере выгода очевидна.

**Пример 3.2:** Вызов функции с фильтрацией рез-та с более сложным условием

Ограничим выдачу запроса 2 строчками

Вызываем простую функцию ------- отработало за 5 секунд

Вызываем конвейерную функцию ---------- отработало за 5 секунд

*Почему в этом кейсе конвейерная функция повела себя как обычная?*

Для того, чтобы отфильтровать по условию in(1, 3) нужно получить весь результат работы.

Потому что не знает, какие элементы в коллекции будут до того, как будет получен весь результат

Поэтому отдаётся всё, а затем уже фильтруется