开发技术:关于变量、移动以及含义(4) PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/143/2021_2022__E5_BC_80_E5_8F_91_E6_8A_80_E6_c102_143239.htm i will assume in my answer that your question refers to server-side-only code that needs to traverse a cursors result set. an entirely different set of considerations comes into play if you are talking about transferring data from the server to a user interface, such as a web page, and allowing that front-end environment to flexibly move through result sets. oracle does not currently support bidirectional access to cursor result sets (aka scrollable cursors) through a pl/sql interface. you might well find, however, that you can achieve the desired effect with a combination of the following: multiple queries (each with different order by clauses that correspond to the different ways you need to traverse the result set). analytic functions: as the oracle database sql reference states, "analytic functions compute an aggregate value based on a group of rows. they differ from aggregate functions in that they return multiple rows for each group. the group of rows is called a window and is defined by the analytic_clause. for each row, a sliding window of rows is defined. the window determines the range of rows used to perform the calculations for the current row. . . ." for tables with a relatively small number of rows, the use of multiple queries may yield a satisfactory implementation. if, on the other hand, your result set is very large, you may run into some performance issues. in addition, you may still not be able to reference arbitrary rows within the result set as desired. fortunately, you can achieve the desired effect of a bidirectional cursor rather easily by caching the result in a pl/sql collection. once the data has been moved into the cache, you can move back and forth through the result set, compare rows, and so on, with complete freedom and a high degree of efficiency. i will demonstrate how you can build and move through such a cache. recall that pl/sql program data consumes program global area (pga) memory, distinct from the system global area (sga), and there is a separate pga for each session connected to an oracle instance. with large result sets, you are going to be manipulating lots of data and the pga will require lots of memory for the collection. this technique of building and moving through a pl/sql collection cache will make the most sense under the following circumstances: you are running this program for a small number of simultaneous sessions, or it is a single batch process. you must have sufficient memory to hold the cache(s) you will create to emulate bidirectional cursors. the data in the result set is static (or you want to ignore any changes that occur once your program starts). once you have copied your result set to your collection-based cache, any changes to the tables that contributed to your result set will not be reflected in the cacheeven if those changes are committed in some other session. this is a "one-off," static copy of the table (or whatever result set you have defined with your query). listing 5 offers an example of bidirectional cursor processing built around a collection of records with the same structure (and data) as the jokes table defined below: create table jokes (joke_id integer, title varchar2(100), text varchar2(4000))/ 100Test 下载频道开通，各