Difference between Regular Cursor and Reference Cursor

|  |  |
| --- | --- |
| **Regular Cursor** | **Reference Cursor** |
| Regular cursor is static and defined at compile time | Reference cursor is dynamic and defined at run time |
| We cannot return regular cursor to the caller | We can return reference cursor to the caller |
| Regular cursor can be used in cursor for loop | Reference cursor can be used in simple for loop, not in cursor for loop |
| Regular cursor can be declared outside of function or procedure as a global cursor | Reference cursor can be declared only inside of same program as a local cursor |
| Regular cursor will be cached in a memory and so CPU will be utilized | Reference cursor will not be cached |

Difference between Static SQL and Dynamic SQL

|  |  |
| --- | --- |
| **Static SQL** | **Dynamic SQL** |
| Static SQL will be compiled at compile time | Dynamic SQL will be compiled at run time |
| Performance is good in Static SQL | Performance is poor comparing to static SQL |
| Limitation is, hard-code is possible as this cannot be edited at run time | Limitation is, there won’t be any hardcode. But if user is not good in SQL, it will be dangerous |
| Its less flexible | Its high flexible |
| This can be used when we are distributing common data | This can be used when we have a change in format of data in distribution |
| EXECUTE IMMEDIATE, USING clause are not needed to access the SQL query | EXECUTE IMMEDIATE, USING clause needed to access the SQL query |
| This is more faster and efficient | This is less efficient |

What is Inline Pragma?

Inline pragma is used for optimization in calling the subprogram. We can switch off or on for inline the subprogram. Let say, we are invoking a subprogram in a loop. If we use pragma inline, Inlining replaces a subprogram invocation with a copy of the invoked subprogram (if the invoked and invoking subprograms are in the same program unit). So performance will be high. It will complete faster than regular invocation without inlining.

What are the levels in PLSQL\_OPTIMIZE\_LEVEL for pragma inline?

* Level 0: no compiler optimizations (PL/SQL compiled as is);
* Level 1: high-level optimizations (such as moving constants out of loops);
* Level 2: default level. Aggressive optimizations (such as rewriting cursor-for-loops as array fetches) and in 11g, also inlining any subprograms that we request with PRAGMA INLINE;
* Level 3: most aggressive level: New in 11g, these will inline all subprograms where possible (excluding those contained in built-in packages).

Refer Link: <http://oraclebymahendra.blogspot.com/2013/11/pragma-inline-in-oracle-11g.html>