**Scrollable cursor**

For a scrollable cursor, the rows of the result table can be fetched many times. The cursor is moved through the result table based on the position option specified on the FETCH statement. When the cursor is opened, it is positioned before the first row in the result table. When a FETCH is issued, the cursor is positioned to the row in the result table that is specified by the position option. That row is then the current row. If host variables are specified (with the INTO clause on the FETCH statement), SQL moves the current row's contents into your program's host variables. Host variables cannot be specified for the BEFORE and AFTER position options.

This sequence is repeated each time a FETCH statement is issued. The cursor does not need to be closed when an end-of-data or beginning-of-data condition occurs. The position options enable the program to continue fetching rows from the table.

The following scroll options are used to position the cursor when issuing a FETCH statement. These positions are relative to the current cursor location in the result table.

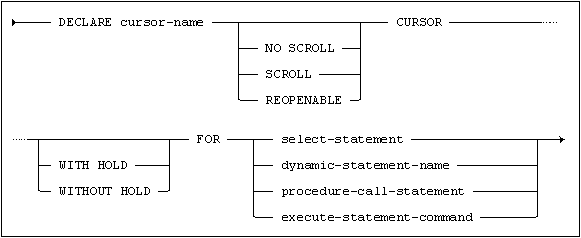
|  |  |
| --- | --- |
| NEXT | Positions the cursor on the next row. This is the default if no position is specified. |
| PRIOR | Positions the cursor on the previous row. |
| FIRST | Positions the cursor on the first row. |
| LAST | Positions the cursor on the last row. |
| BEFORE | Positions the cursor before the first row. |
| AFTER | Positions the cursor after the last row. |
| CURRENT | Does not change the cursor position. |
| RELATIVE n | Evaluates a host variable or integer *n* in relationship to the cursor's current position. For example, if *n* is -1, the cursor is positioned on the previous row of the result table. If *n* is +3, the cursor is positioned three rows after the current row. |

For a scrollable cursor, the end of the table can be determined by the following:

**FETCH AFTER FROM** C1

Once the cursor is positioned at the end of the table, the program can use the PRIOR or RELATIVE scroll options to position and fetch data starting from the end of the table.

Declares a cursor definition.



### Usage

Embedded/Procedural.

### Description

A cursor is declared in accordance with the select-statement or the result set procedure call specified in procedure-call-statement.

The select-statement may be specified explicitly in ordinary embedded SQL applications or by the name of a prepared SELECT, identified by dynamic-statement-name, in dynamic SQL statements, see the [Mimer SQL Programmer's Manual, Dynamic SQL](http://developer.mimer.com/documentation/html_92/Mimer_SQL_Engine_DocSet/Dynamic_SQL.html#wp1114954).

The cursor is identified by cursor-name, and may be used in FETCH, DELETE CURRENT and UPDATE CURRENT statements. The cursor must be activated with an OPEN statement before it can be used.

A cursor declared as REOPENABLE may be opened several times in succession, and previous cursor states are saved on a stack, see [OPEN](http://developer.mimer.com/documentation/html_92/Mimer_SQL_Engine_DocSet/SQL_Statements63.html#wp1117397). Saved cursor states are restored when the current state is closed, see [CLOSE](http://developer.mimer.com/documentation/html_92/Mimer_SQL_Engine_DocSet/SQL_Statements13.html#wp1113136).

A cursor declared as SCROLL will be a scrollable cursor. For a scrollable cursor, records can be fetched using an orientation specification. See the description of [FETCH](http://developer.mimer.com/documentation/html_92/Mimer_SQL_Engine_DocSet/SQL_Statements51.html#wp1401851) for a description of how the orientation can be specified.

A cursor declared WITH HOLD will be a holdable cursor. Open holdable cursors are not closed when a transaction is committed. A cursor will be non-holdable if WITHOUT HOLD is explicitly specified.

WITHOUT HOLD and NO SCROLL are default cursor attributes and do not have to be specified

//////////////////////////////////////////////

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Abstract** | | Learn how you can write powerful applications by having your cursor scroll though your DB2® for z/OS® tables -- fetch forwards, backwards or jump around. See how this function fits in with the ODBC/JDBC environment and how you can use it from different platforms. | |  | |  | |
|  |
| **Content** |
| Use scrollable cursors in screen-based applications to allow the user to scroll backwards and forwards through the result table or in decision support applications to explore large result tables through a cursor. Implement your scrollable cursors to be insensitive to changes or sensitive to changes. You can even make your scrollable cursors updatable. This presentation helps you get acquainted with ways to navigate through your work horse DB2 table while connected through a user friendly workstation.  Specifically, this presentation provides information about the following topics:   * Implementing Scrollable Cursors Across Platforms * Scrollable Cursor Features * Updating Using Scrollable Cursors * Scrollable Cursors Compatibility Across Existing Functions * Performance Considerations Using Scrollable Cursors |