crystal decisions™

Crystal Reports

Cursors in Microsoft SQL Server

Overview

This document will define what a cursor is, how to use a cursor, the purpose of a cursor, and various cursor commands.

Contents

Introduction

Server cursors allow individual record operations to be performed on a given results set or on the entire set. Server cursors enhance the general use of relational databases by allowing record-level operations when set-oriented operations are impractical.

Using cursors, multiple operations can be performed record by record against a results set with or without returning to the original table. In other words, cursors conceptually return results set based on tables within the database(s).

For example, a cursor can be generated to include a list of all user-defined table names within a database. After the cursor has been opened movement (fetching) through the results set can include multiple operations against each table by passing each table name as a variable.

Cursors are extremely powerful when combined with stored procedures and the EXECUTE statement (to dynamically build strings). Also, each FETCH statement will return only one row from the results set. In relational systems, set operations are preferred; cursors provide a powerful complement to set operations

SQL Server provides two interfaces for cursor functions

When using cursors in Transact-SQL batches or stored procedures, ANSI-standard SQL syntax has been added for declaring, opening, and fetching from cursors as well as positioned updates and deletes.

- When using cursors from a DB-Library or ODBC program, the SQL Server 6.0 client libraries transparently call built-in server functions to handle cursors more efficiently.

The overview below shows the statements, in order, used to implement a cursor.

Statement	**Description**
DECLARE CURSOR Statement	Defines a cursor.
OPEN Statement	Opens a declared cursor.
FETCH Statement	Retrieves a specific row from the cursor.
CLOSE Statement	Closes an open cursor.
DEALLOCATE Statement	Removes the cursor data structures.

In addition to browsing through the cursor results with FETCH, the current row of an open cursor can be modified. The UPDATE statement and the DELETE statement, when used against a cursor, affect only the row on which the cursor is currently positioned.

Syntax

To update the current row:

```
UPDATE table_name
       SET column_name1 =
                      {expression1 | NULL |
(select_statement)}
                      [, column_name2 =
                      {expression2 | NULL |
(select_statement)}...]
WHERE CURRENT OF cursor_name
```

To delete the current row:

```
DELETE FROM table_name

WHERE CURRENT OF cursor_name
```

Modifications made with WHERE CURRENT OF cursor_name affect only the single row on which the cursor is positioned. In the case of a cursor based on a join, only the table_name specified in the DELETE or UPDATE statement is modified. Other tables participating in the cursor are not affected. For complete syntax and more information, see the DELETE and UPDATE statements.

The following sections discuss each of the statements used to define, access, and close a cursor.

Declare statement

Defines a cursor. Once declared, row sets for cursors are assigned by using the OPEN statement. (The DECLARE statement can also be used to define the name and type of local variables for a batch or procedure. For more information on variable declaration and assignment, see the DECLARE statement.)

Syntax

DECLARE cursor_name [INSENSITIVE] [SCROLL] CURSOR
FOR select_statement
[FOR {READ ONLY | UPDATE [OF column_list]}]

WHERE

cursor_name

This is the name of the cursor being defined. The cursor_name must conform to the rules for identifiers.

INSENSITIVE

Defines a cursor that makes a temporary copy of the data to be used by the cursor. All requests to the cursor are answered from this temporary table; therefore, modifications made to base tables will not be reflected in the data returned by fetches made to this cursor, and this cursor does not allow modifications.

SCROLL

Specifies that all methods of fetching data are available. Committed deletes and updates made to the underlying tables (by any users) are reflected in subsequent fetches (provided that the cursor is not declared with the INSENSITIVE option).

select_statement

A standard SELECT statement used to identify a set of rows from a given table or tables within a database. If each of the underlying tables does not have a unique index, the cursor will automatically be an INSENSITIVE cursor.

The keywords COMPUTE, COMPUTE BY, FOR BROWSE, and INTO are not allowed within the select_statement of a cursor declaration.

If DISTINCT, UNION, GROUP BY, and/or HAVING are used, one or more of the underlying tables does not have a unique index, an outer join is used, or a constant expression is included in the select_list, the cursor will be created as INSENSITIVE.

READ ONLY

Prevents updates from occurring against any row within this cursor. This option overrides the default capability of a cursor to be updated.

UPDATE [OF column_list]

Defines updateable columns within the cursor. If OF column_list is supplied, only the columns listed will allow modifications. If no list is supplied, all columns can be updated unless the cursor has been defined as READ ONLY.

OPEN Statement

Opens a cursor.

Syntax

OPEN cursor_name

WHERE

cursor_name

Specifies the name of an already-declared cursor. In the case of an INSENSITIVE cursor, this will generate the required temporary table. At the time the cursor is opened, results set membership and ordering are fixed.

NOTE	Membership and ordering are fixed when the cursor is opened; updates and deletes that have been committed against the base tables of the cursor by other users will be reflected in fetches made against all cursors defined without the INSENSITIVE option. This behavior is commonly called keyset-driven. Rows that have been deleted from a results set will appear with NULLs in all variable columns, spaces in all char columns, and default or zero values in all other non-null columns. To differentiate between a deleted row and a row that has been inserted with NULL data values, use the global variable @@FETCH_STATUS after each FETCH is made against the cursor.

Remarks

Once a cursor has been opened, use the global variable @@CURSOR_ROWS to receive the number of qualifying rows in the last opened cursor. Depending on the number of rows expected in the results set, SQL Server might choose to populate the keyset asynchronously on a separate thread. This allows fetches to proceed immediately, even if the keyset is not fully populated. @@CURSOR_ROWS returns:

–m	If the cursor is being populated asynchronously. The value returned (–m) refers to the number of rows currently in the keyset.
n	If the cursor is fully populated. The value returned (n) refers to the number of rows.
0	If no cursors have been opened or the last opened cursor has been closed or deallocated.

To set the threshold at which SQL Server will generate keysets asynchronously, use the **cursor threshold** configuration option. For details, see the **sp_configure** system stored procedure.

FETCH Statement

Retrieves a specific row from the cursor.

Syntax

FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE n | RELATIVE n]
FROM] cursor_name
[INTO @variable_name1**,** @variable_name2**,** ...]

WHERE

NEXT

Returns the first row of the results set if this is the first fetch against the cursor; otherwise, it moves the cursor one row within the results set. NEXT is the primary method used to move through a results set. NEXT is the default cursor fetch.

NOTE	PRIOR, FIRST, LAST, ABSOLUTE n, and RELATIVE n are available only with cursors defined with the SCROLL option.

PRIOR

Returns the previous row within the results set.

FIRST

Moves the cursor to the first row within the results set and returns the first row.

LAST

Moves the cursor to the last row within the results set and returns the last row.

ABSOLUTE n

Returns the *n*th row within the results set. If n is a negative value, the returned row will be the nth row counting backward from the last row of the results set.

RELATIVE n

Returns the nth row after the currently fetched row. If n is a negative value, the returned row will be the nth row counting backward from the relative position of the cursor.

FROM cursor_name

Defines the cursor from which the fetch should be made. Multiple cursors are allowed within any session provided that each has a unique name.

INTO @variable_name1, @variable_name2, ...

Allows data returned within a fetch to be placed into local variables. Each of the variables must match the datatype. Errors will occur when the datatypes are incompatible. Implicit datatype conversions are not provided here.

A global variable, @@FETCH_STATUS*,* will be updated at every execution of FETCH. At a successful fetch, @@FETCH_STATUS will be set to 0. If no data was fetched because the requested cursor position exceeded the results set, –1 will be returned. If the row returned is no longer a member of the results set (for example, the row was deleted from the base table after the cursor was opened), @@FETCH_STATUS will return –2. Always use this to determine the validity of the data returned from a cursor fetch prior to attempting any operation against that data.

For more information about how SQL Server 6.5 affects existing database systems and for updated SQL Server 6.5 syntax, see the FETCH Statement in your SQL Server documentation.

CLOSE Statement

Closes an open cursor. CLOSE leaves the data structures accessible for reopening; however, modifications or fetches are not allowed until the cursor is reopened.

Syntax

CLOSE cursor_name

WHERE

cursor_name

Specifies the name of an already-declared cursor.

DEALLOCATE Statement

Removes the cursor data structures. DEALLOCATE is different from CLOSE in that a closed cursor can be re-opened. DEALLOCATE releases all data structures associated with the cursor and removes the definition of the cursor.

Syntax

DEALLOCATE cursor_name

WHERE

cursor_name

Specifies the name of an already-declared cursor.

Examples

Simple cursor and syntax

The results set generated at the opening of this cursor includes all rows and all columns in the Authors table of the Pubs database. This cursor can be updated, and all updates and deletes will be represented in fetches made against this cursor. FETCH NEXT is the only fetch available because the SCROLL option has not been specified.

```
DECLARE authors_cursor CURSOR

        FOR SELECT * FROM authors

OPEN authors_cursor

FETCH NEXT FROM authors_cursor
```

Contacting Crystal Decisions for Technical Support

We recommend that you refer to the product documentation and that you visit our Technical Support web site for more resources.

Self-serve Support:

http://support.crystaldecisions.com/

Email Support:

http://support.crystaldecisions.com/support/answers.asp

Telephone Support:

http://www.crystaldecisions.com/contact/support.asp