NAME

SPI_cursor_open - set up a cursor using a statement created with **SPI_prepare**

SYNOPSIS

Portal SPI_cursor_open(const char * *name*, SPIPlanPtr *plan*,
Datum * *values*, const char * *nulls*,
bool *read_only*)

DESCRIPTION

SPI_cursor_open sets up a cursor (internally, a portal) that will execute a statement prepared by **SPI_prepare**. The parameters have the same meanings as the corresponding parameters to **SPI_execute_plan**.

Using a cursor instead of executing the statement directly has two benefits. First, the result rows can be retrieved a few at a time, avoiding memory overrun for queries that return many rows. Second, a portal can outlive the current C function (it can, in fact, live to the end of the current transaction). Returning the portal name to the C function's caller provides a way of returning a row set as result.

The passed-in parameter data will be copied into the cursor's portal, so it can be freed while the cursor still exists.

ARGUMENTS

const char * *name*
name for portal, or NULL to let the system select a name

SPIPlanPtr *plan*
prepared statement (returned by **SPI_prepare**)

Datum * *values*
An array of actual parameter values. Must have same length as the statement's number of arguments.

const char * *nulls*
An array describing which parameters are null. Must have same length as the statement's number of arguments.

If *nulls* is NULL then **SPI_cursor_open** assumes that no parameters are null. Otherwise, each entry of the *nulls* array should be ' ' if the corresponding parameter value is non-null, or 'n' if the corresponding parameter value is null. (In the latter case, the actual value in the corresponding *values* entry doesn't matter.) Note that *nulls* is not a text string, just an array: it does not need a

’\0’ terminator.

bool *read_only*
true for read-only execution

RETURN VALUE

Pointer to portal containing the cursor. Note there is no error return convention; any error will be reported via **elog**.