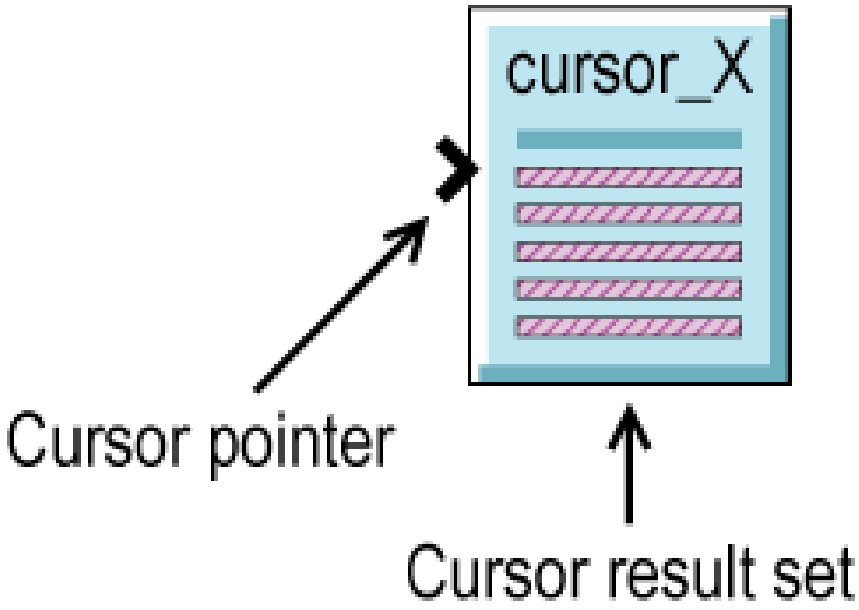
CURSORES

Preparó: Ismael Castañeda Fuentes
Fuentes: Manuales Sybase
Manuales Oracle

Cursor

Un cursor es un mecanismo que sirve para procesar fila por fila los resultados de una consulta

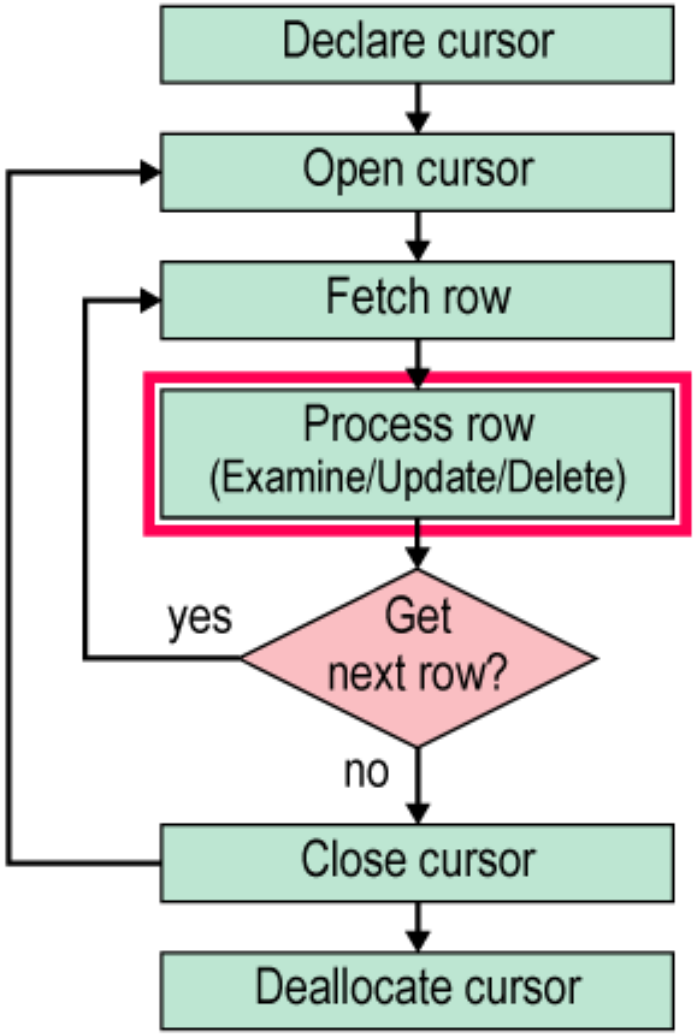


Beneficios de los cursores

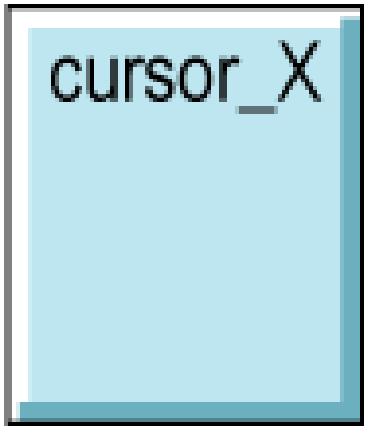
- Se pueden procesar los datos fila por fila
  - SQL es un lenguaje orientado a conjuntos
  - El procesamiento se hace normalmente sobre las filas que cumplan con una condición dada
  - Los cursors permiten el procesamiento fila por fila
- Se pueden modificar los datos fila por fila
- Se puede sortear la brecha existente entre la orientación a conjuntos de las bases de datos relacionales y la orientación a filas de muchos lenguajes de programación

Ciclo de vida de un cursor

1. Declarar el cursor
2. Abrir el cursor
3. Tomar cada fila
4. Cerrar el cursor
5. Desasignar el cursor



Paso 1: Declarar el cursor



Cuando se declara un cursor:

- Se especifica una consulta
- Se especifica un modo para el cursor
  - **De solo lectura**
  - **Para actualización**

Sintaxis para declarar un cursor

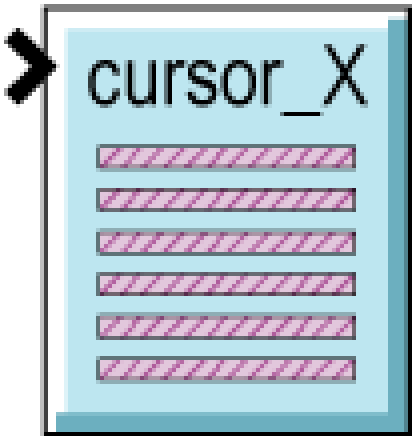
- Sintaxis simplificada:

  declare *cursor_name* **cursor**
  for *select_statement*
  [**for** { **read only** | **update** [of *column_name_list*] }]

- Ejemplo:

```
declare biz_book cursor
    for select title, title_id from titles
          where type = "business"
    for read only
go
```

Paso 2: Abrir el cursor



- Cuando se abre el cursor
  - El servidor crea el conjunto resultado
  - El apuntador está señalando antes de la primera fila del conjunto respuesta

Sintaxis para la apertura de un cursor

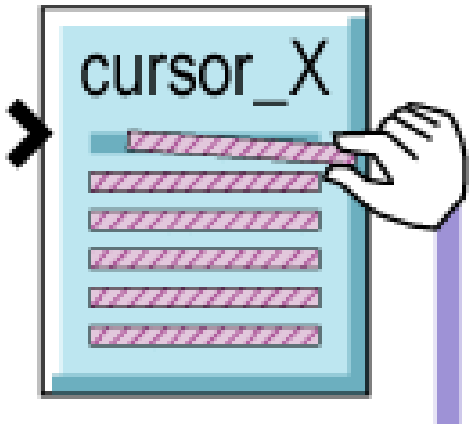
- Sintaxis:

  open *cursor_name*

- Ejemplo:

```
declare biz_book cursor
     for select title, title_id from titles
           where type = "business"
     for read only
go
declare @title char(80), @title_id char(6)
open biz_book
fetch biz_book into @title, @title_id
while @@sqlstatus = 0
     begin
           -- process @title and @title_id
           fetch biz_book into @title, @title_id
     end
close biz_book
deallocate cursor biz_book
```

Paso 3: Tomar cada fila



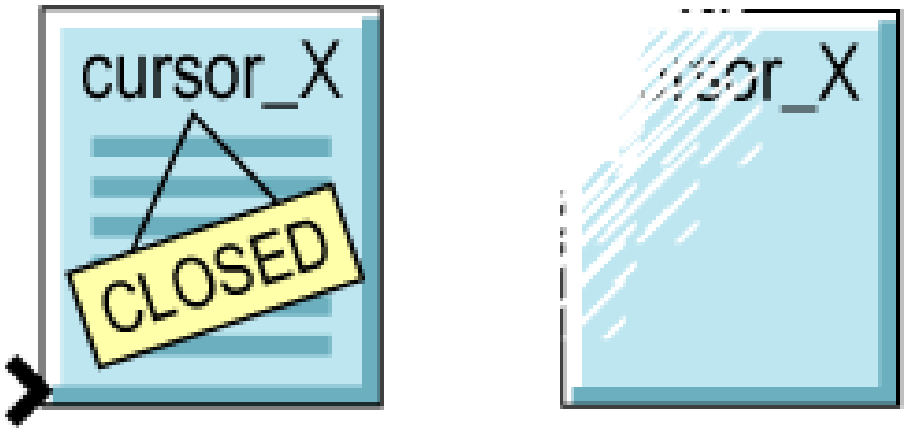
- Cuando se ejecuta un **fetch**:
  - El cursor señala a la siguiente fila válida
  - Retorna la siguiente fila válida

Sintaxis de un fetch

- Sintaxis:
  fetch *cursor_name* [**into** *fetch_target_list*]
- Ejemplo:

```
declare biz_book cursor
     for select title, title_id from titles
                    where type = "business"
              for read only
go
declare @title char(80), @title_id char(6)
open biz_book
fetch biz_book into @title, @title_id
while @@sqlstatus = 0
     begin
              -- process @title and @title_id
                    fetch biz_book into @title, @title_id
     end
close biz_book
deallocate cursor biz_book
```

Pasos 4 y 5: Cerrar y desasignar el Cursor



- Cuando se cierra un cursor:
  - Termina el procesamiento de la consulta hecha
- Cuando se desasigna el cursor:
  - Se liberan todos los recursos de memoria asignados al cursor

Cerrar y desasignar un Cursor

- Sintaxis:
  close *cursor_name*
  deallocate cursor *cursor_name*
- Ejemplo:

```
declare biz_book cursor
     for select title, title_id from titles
            where type = "business"
     for read only
go
declare @title char(80), @title_id char(6)
open biz_book
fetch biz_book into @title, @title_id
while @@sqlstatus = 0
     begin
            -- process @title and @title_id
            fetch biz_book into @title, @title_id
     end
close biz_book
deallocate cursor biz_book
```

Variables para el manejo de cursores

- Se tiene una variable que retorna el número total de filas procesadas (*@@rowcount)*
- Se tiene una variable que indica el estado o resultado de mover el cursor (*@@sqlstatus)*
  - Exitoso: se alcanzó una fila válida
  - Hay un error al tratar de tomar la fila
  - Ya se procesaron todas las filas

Notas adicionales para fetch

- **fetch** siempre mueve el apuntador a la siguiente fila válida en el conjunto respuesta
  - Algunos servidores permiten regresarse a una fila anterior
  - Cerrar y reabrir un cursor hace que el apuntador siempre señale al comienzo
- Por default, **fetch** siempre retorna una fila
  - Algunos servidores permiten cambiar este defaullt
    - Sintaxis:
      set cursor rows *number* **for** *cursor_name*
    - Ejemplo:
      `set cursor rows 5 for biz_book`

Prácticas recomendadas para desarrollo

- Siempre especificar el modo del cursor en la sentencia **declare**
- Como los cursores pueden demandar muchos recursos, evitar dejar abiertos los cursores por mucho
- Si se ejecuta la misma operación en cada fila del cursor, hay que buscar una alternativa

Ejemplo de cursor 1/3

```
declare books_csr cursor for
select title_id, type, price
from titles
for read only
go

-- List all business and mod_cook books. Show business
  books
-- at 8% increase in price. This cursor allows you to
-- selectively manipulate a subset of the rows while
-- retaining a single result set.
declare     @title_id          tid,
            @type              char(12),
            @price                   money
open books_csr
-- initial fetch
fetch books_csr into @title_id, @type, @price
```

Ejemplo de cursor 2/3

```
while @@sqlstatus = 0
  begin
     if @@sqlstatus = 1
        begin
           raiserror 30001 "select failed"
           close books_csr
           deallocate cursor books_csr
           return
        end
     if @type="business"
           select @title_id, @type,CONVERT(money,@price*1.08)
        else
           if @type="mod_cook"
              select @title_id, @type, @price
     -- subsequent fetches within loop
     fetch books_csr into @title_id, @type, @price
  end
```

Ejemplo de cursor 3/3

```
close books_csr
deallocate cursor books_csr
go
```

- Results:

```
-      -            ------------------------
BU1032 business                      21.59
-      -            ------------------------
BU1111 business                      12.91
-      -            ------------------------
BU2075 business                       3.23
-      -            ------------------------
BU7832 business                      21.59
-      -            ------------------------
MC2222 mod_cook                      19.99
-      -            ------------------------
MC3021 mod_cook                       2.99
```

Alternativas al uso de cursores

- Los cursores no son la única manera de ejecutar una tarea
- Alternativa: usar **case**

```
select title_id, type,
     case type
            when "business" then price * $1.08
            when "mod_cook" then price
            end
     from titles
     where type in ("business", "mod_cook")
```

- Alternativa: hacer dos consultas:

```
select title_id, type, price * $1.08
     from titles
     where type = "business"
select title_id, type, price
     from titles
     where type = "mod_cook"
```

Ejemplo de cursor 1/3

```
declare title_author_csr cursor for
  select authors.au_id, au_fname, au_lname, title
  from titles, authors, titleauthor
  where titles.title_id = titleauthor.title_id
  and authors.au_id = titleauthor.au_id
  order by upper(au_lname), upper(au_fname)
  for read only
go

set nocount on --Turns off display of rows affected
declare @fname varchar(20), @lname varchar(40),
        @title varchar(80), @au_id char(11),
        @old_au_id char(11)
open title_author_csr
fetch title_author_csr into @au_id, @fname, @lname,
  @title
```

Ejemplo de cursor 2/3

```
while @@sqlstatus = 0
  begin
     if @@sqlstatus = 1
       begin
         raiserror 23000 "Select failed."
         return
       end
     if @au_id <> @old_au_id
       begin
         print "   "
         print "%1! %2! is the author of these books:",
          @fname, @lname
       end
     print "    %1!", @title
     select @old_au_id = @au_id
     fetch title_author_csr into @au_id, @fname, @lname,
          @title
  end
```

Ejemplo de cursor 3/3

```
close title_author_csr
deallocate cursor title_author_csr
set nocount off --Turns back on display of rows affected
go
```

- Resultados:

```
...
Ann Dull is the author of these books:
       Secrets of Silicon Valley

Marjorie Green is the author of these books:
       You Can Combat Computer Stress!
       The Busy Executive's Database Guide

Burt Gringlesby is the author of these books:
       Sushi, Anyone?
...
```

Actualizar datos usando cursores

- Sintaxis simplificada:

  update *table_name*
  set *column1* = { *expression* | *select_statement* }
  [, *column2* = { *expression* | *select_statement* } ...]
  where current of *cursor_name*

- Ejemplo:

```
update titles
    set title = "The Executive's Database Guide"
    where current of biz_book
```

- Actualiza la fila a la que señala el apuntador
  - En la mayoría de casos, esta fila es la tomada más recientemente
- **NO** mueve el cursor a la siguiente fila
- Sólo se pueden actualizar cursores declarados en modo **update**

Borrar datos usando cursores

- Sintaxis simplificada:
  delete [**from**] *table_name* **where current of** *cursor_name*
- Ejemplo:

```
delete from titles
where current of biz_book
```

- Borra la fila que está siendo señalada por el apuntador
  - En la mayoría de casos, esta fila es la tomada más recientemente
- Mueve el apuntador del cursor a la fila siguiente
- Sólo se pueden actualiza cursores declarados en modo **update**

Reglas para actualizar cursores

- La tabla sobre la cual el cursor va a actuar debe estar declarada:
  - Con un índice único

    o
  - Usando un esquema de bloqueo tipo Datapages o Datarows

Ejemplo de cursor 1/3

```
-- Increase all prices less than the average price by
  50%
-- Decrease all prices greater than or equal to the
  average
--  price by 25%
declare title_update cursor
for select title_id, price from titles
for update
declare     @avg_price money,        -- local variables
            @title_id  tid,
      @price     money
open title_update                    -- execute cursor
begin tran
-- calculate average price
select @avg_price = avg(price) from titles holdlock
fetch title_update into @title_id, @price
```

Ejemplo de cursor 2/3

```
while @@sqlstatus = 0
  begin
     if @@sqlstatus = 1                -- error occurred
       begin
          rollback tran
          raiserror 21001 "Fetch failed in cursor"
          close title_update
          deallocate cursor title_update
          return
       end
     if @price < @avg_price
          update titles                --increase by 50%
             set price = price * $1.50
             where current of title_update
       else
          update titles                -- decrease by 25%
             set price = price * $.75
             where current of title_update
```

Ejemplo de cursor 3/3

```
    if @@error <> 0
       begin
           rollback tran
           raiserror 22001 "Update failed"
           close title_update
           deallocate cursor title_update
           return
       end
  fetch title_update into @title_id, @price
end
commit tran
close title_update
deallocate cursor title_update
go
...
```

Cursores y transacciones

- Para cursores **for update** obtener bloqueos update
  - Los bloqueos se promueven a bloqueos exclusivos cuando se ejecuta un **update where current of** o **delete where current of**
  - Si no se promueve, el bloqueo update se libera cuando el cursor se mueve a la siguiente página de datos
- **close on endtran** es una opción que determina qué le pasa al cursor en una transacción cuando se llega a un **rollback** o **commit**
  - Cuando está activo, el cursor se cierra después de un **rollback** o **commit**
  - Cuando no está activo:
    - El cursor permanece abierto después de un **rollback** o **commit**
    - Las modificaciones basadas en la posición de un cursor se pueden ejecutar fila por fila, lo cual puede incrementar la concurrencia
  - Sintaxis:
    set close on endtran { on | off }

Cursor a nivel de servidor

- Un cursor a nivel de servidor es aquel creado en un stored procedure
- Ejemplo:

```
create proc proc_fetch_book
as
declare    @title                char(30),
           @title_id             char(6)
declare biz_book cursor
    for select title, title_id from titles
           where type = "business"
    for read only
open biz_book
fetch biz_book into @title, @title_id
-- additional processing here
close biz_book
deallocate cursor biz_book
return
```

Alcance de cursores a nivel servidor

- Los “stored procedures” pueden tomar datos de cursores creados por un procedimiento que llama al procedimiento dado