

Base de Datos II

Facilitador: César Bustamante Chong
cbustamante@ecotec.edu.ec

AGENDA

SENTENCIAS TRANSACT SQL

➤ CURSORES



- **AGENDA**
 - INTRODUCCION A TRANSACT SQL
 - CURSORES

INTRODUCCION A CURSORES

- **¿Que es un Cursor?**
- **CURSOR** se refiere a una estructura de control utilizada para el recorrido (y potencial procesamiento) de los registros del resultado de una **consulta**.
- Un cursor se utiliza para el procesamiento individual de las filas devueltas por el sistema gestor de base de datos para una consulta.
- Es necesario debido a que muchos lenguajes de programación sufren de lo que en inglés se conoce como **impedance mismatch**.
- Por norma general los lenguajes de programación son procedurales y no disponen de ningún mecanismo para manipular conjuntos de datos en una sola instrucción.
- Debido a ello, las filas deben ser procesadas de forma secuencial por la aplicación.

INTRODUCCION A CURSORES

- **¿Que es un Cursor?**
- Un cursor puede verse como un [iterador](#) sobre la colección de filas que habrá en el set de resultados.
- Existen sentencias [SQL](#) que no requieren del uso de cursores. Ello incluye la sentencia [Insert](#), así como la mayoría de formas del [Update](#) o el [Delete](#).
- Incluso una sentencia [Select](#) puede no requerir un cursor si se utiliza en la variante de `SELECT...INTO`, ya que esta variante sólo devuelve una fila.

INTRODUCCION A CURSORES

- **COMPONENTES DE UN CURSOR**
- Un cursor es creado utilizando la sentencia DECLARE CURSOR. Es obligatorio asignarle un nombre.

DECLARE cursor_name CURSOR FOR SELECT... FROM...

- Antes de ser utilizado, el cursor debe ser abierto con una sentencia OPEN. Como resultado de esta sentencia, el cursor se posiciona *antes* de la primera fila del set de resultados

OPEN cursor_name

INTRODUCCION A CURSORES

- **COMPONENTES DE UN CURSOR**
- Un cursor se posiciona en una fila específica del set de resultados con la sentencia FETCH.
- Una sentencia fetch transfiere la información de la fila a la aplicación.
- Una vez todas las filas han sido procesadas o la sentencia fetch queda posicionada en una fila no existente, el SGBD devuelve un SQLSTATE '02000' (acompañado normalmente de un SQLCODE +100) para indicar el final del set de resultados.

FETCH *cursor_name* INTO...

INTRODUCCION A CURSORES

- **COMPONENTES DE UN CURSOR**
- El último paso consiste en cerrar el cursor utilizando la sentencia CLOSE.

CLOSE *cursor_name*

IMPORTANTE: Una vez un cursor está cerrado puede reabrirse de nuevo, lo cual implica que la consulta es reevaluada y se crea un nuevo set de resultados

INTRODUCCION A CURSORES

Cursores de recorrido

- Los cursores pueden declararse como de recorrido o no.
- Si son de recorrido, éste indica la dirección en la que el cursor puede moverse.
- Un cursor **sin recorrido** (non-scrollable en inglés) también se conoce como cursor **unidireccional** (forward-only en inglés).
- Cada fila puede ser leída como mucho una vez, y el cursor automáticamente se mueve a la siguiente fila.
- Una operación de fetch después de haber recuperado la última fila posiciona el cursor detrás de la misma y devuelve SQLSTATE 02000 (SQLCODE +100).
- Un cursor de recorrido puede posicionarse en cualquier parte del set de resultados utilizando la sentencia SQL FETCH.
- La palabra clave debe ser especificada cuando se declare el cursor. El valor por defecto es NO SCROLL.

INTRODUCCION A CURSORES

- **COMPONENTES DE UN CURSOR DE RECORRIDO**
- Declarar que es un cursor de recorrido

DECLARE *cursor_name* sensitivity **SCROLL** CURSOR FOR SELECT... FROM...

- La posición de un cursor de recorrido puede especificarse de forma relativa a la posición actual del cursor o de forma absoluta a partir del principio del set de resultados.

FETCH [NEXT | PRIOR | FIRST | LAST] FROM *cursor_name*
FETCH ABSOLUTE *n* FROM *cursor_name*
FETCH RELATIVE *n* FROM *cursor_name*

INTRODUCCION A CURSORES

Cursores de recorrido

- Los cursores de recorrido pueden potencialmente acceder a la misma fila del set de resultados múltiples veces.
- Por lo tanto, modificaciones de datos (insert, update, delete) realizadas por otras transacciones podrían tener un impacto en el set de resultados.
- Un cursor puede ser **sensible** o **insensible** a tales modificaciones.
- Un cursor sensible recogería las modificaciones que afectarían al set de resultados, mientras que uno insensible no.
- Adicionalmente, un cursor puede ser **asensible**, en cuyo caso el SGBD intentará, en la medida de lo posible, aplicar los cambios como si fuera sensible.

EJEMPLO DE CURSORES

➤ VALIDACIONES

```
if exists (select * from sysobjects where id =  
    object_id(N'[dbo].[nombre_sp]') and OBJECTPROPERTY(id,  
    N'IsProcedure') = 1)  
drop procedure [dbo].[nombre_sp]  
GO
```

```
SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS OFF  
GO
```

EJEMPLO DE CURSORES

➤ CREACION DEL SP Y DECLARAR VARIABLES DEL SP

```
create proc nombre_sp  
    (@i_empresa_id integer,  
     @i_fecha_desde datetime,  
     @i_fecha_hasta datetime,  
     @i_periodo_id smallint,  
     @i_anio_id smallint)  
  
as  
    declare @w_empresa integer,  
            @w_fecha datetime,  
            @w_cliente decimal(5),  
            @w_nombre varchar(100),
```

EJEMPLO DE CURSORES

➤ **COMIENZO DEL SP, CON BEGIN TRAN Y COMMIT TRAN**

BEGIN TRAN

- 1) *DECLARAR VARIABLES*
- 2) *DECLARAR EL CURSOR*
- 3) *OPEN EL CURSOR*
- 4) *FETCH EL CURSOR*
- 5) *LEE EL CURSOR CON WHILE*

- 6) *CLOSE EL CURSOR*
- 7) *DEALLOCATE EL CURSOR*

END TRAN

EJEMPLO DE CURSORES

➤ COMIENZO DEL SP, CON BEGIN TRAN Y COMMIT TRAN

BEGIN TRAN

1)

```
select @w_primer_reg = "S"  
select @w_reg_antes = 0  
select @w_reg_actual = 0  
select @w_total_fac = 0  
select @w_total_fac_S = 0  
select @w_total_pag = 0  
select @w_total_pag_ad = 0  
select @w_total_NC = 0  
select @w_total_ND = 0  
select @w_total_AD = 0  
select @w_total_AC = 0
```

END TRAN

EJEMPLO DE CURSORES

➤ COMIENZO DEL SP, CON BEGIN TRAN Y COMMIT TRAN

BEGIN TRAN

2) DECLARE EL CURSOR

DECLARE cur_selecciona SCROLL CURSOR for

```
select CAB.emp_empresa_id, CAB.anc_fecha_registro, CAB.cli_numero_id, '',  
      DET.cli_distribuidor_id, 0, DET.anc_numero_factura, DET.anc_valor,  
      0, anc_estado_sri  
from mcc_t_aviso_nc_cab CAB, mcc_t_aviso_nc_det DET  
where CAB.emp_empresa_id = @i_empresa_id  
and CAB.anc_fecha_registro >= @i_fecha_desde  
and CAB.anc_fecha_registro <= @i_fecha_hasta  
and CAB.anc_tipo in ('CO')  
and CAB.anc_estado_reg in ('P')  
and CAB.emp_empresa_id = DET.emp_empresa_id  
and CAB.anc_secuencial_id = DET.anc_secuencial_id  
and DET.mmc_motivo_id not in (16, 32)  
order by DET.anc_numero_factura, CAB.cli_numero_id
```

END TRAN

EJEMPLO DE CURSORES

➤ **COMIENZO DEL SP, CON BEGIN TRAN Y COMMIT TRAN**

BEGIN TRAN

3) *OPEN EL CURSOR*

OPEN cur_selecciona

4) *FETCH EL CURSOR*

FETCH cur_selecciona

*INTO @w_empresa, @w_fecha, @w_cliente, @w_nombre,
@w_distribuidor, @w_tipo_cliente, @w_factura,
@w_valor, @w_saldo_final, @w_estado_sri*

END TRAN

EJEMPLO DE CURSORES

➤ COMIENZO DEL SP, CON BEGIN TRAN Y COMMIT TRAN

BEGIN TRAN

5) LEE EL CURSOR CON WHILE

```
WHILE (@@FETCH_STATUS != -1)
BEGIN
    IF (@@FETCH_STATUS != -2)
    BEGIN
        IF not exists(select * from mcc_r_estado_factura_aplicaciones
                        where emp_empresa_id = @i_empresa_id
                        and numero_factura = @w_factura
                        and anio_proceso_id = @i_anio_id
                        and per_periodo_id = @i_periodo_id)
        BEGIN
            select @w_tipo_factura = fac_tipo_factura
            from mfa_t_factura_cab
            where emp_empresa_id = @i_empresa_id
            and fac_factura_id = @w_factura
            if @w_tipo_factura = 1
            BEGIN
                select @w_total_AC = @w_total_AC + @w_valor
            END
        END
    END
```

END

END

END TRAN

EJEMPLO DE CURSORES

➤ **COMIENZO DEL SP, CON BEGIN TRAN Y COMMIT TRAN**

BEGIN TRAN

*5) REALIZA LA ACCION POR EL CUAL EL CURSOR ES
INVOCADO*

```
FETCH cur_selecciona  
INTO @w_empresa, @w_fecha, @w_cliente, @w_nombre,  
      @w_distribuidor, @w_tipo_cliente, @w_factura, @w_valor,  
      @w_saldo_final, @w_estado_sri  
END
```

```
UPDATE mcc_r_estado_factura_aplicaciones  
SET valor_inicial = valor_inicial + @w_total_AC,  
    saldo_inicial = saldo_inicial + @w_total_AC,  
    tot_Avisos_C = @w_total_AC  
WHERE emp_empresa_id = @i_empresa_id  
and per_periodo_id = @i_periodo_id  
and estado_periodo_creacion = 99  
and estado_mes_reg = 'V'
```

END TRAN

EJEMPLO DE CURSORES

➤ **RELIZA EL CLOSE CURSOR, DEALLOCATE**

```
close cur_selecciona  
DEALLOCATE cur_selecciona
```

```
Commit tran  
return 0  
GO  
SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON  
GO
```

- **RELIZA LAS DOS ACTUALIZACIONES SOLICITADAS EN CLASE Y PROCESADOS POR UN CURSOR**