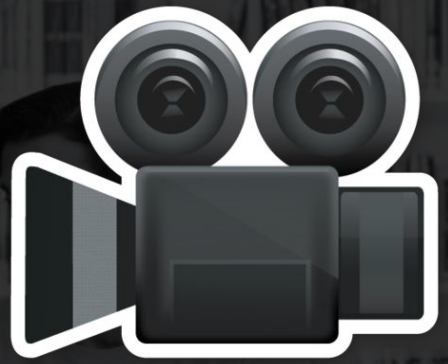
Programador COBOL CICS

Codeki™

Recuerda poner a grabar la clase.




Conceptos adicionales
en CICS

Mapa Conceptual

01
TEMPORARY STORAGE

02
TRANSIENT DATA

03
UNIDAD DE TRABAJO

Agenda

1. Temporary Storage
2. Transient Data
3. Concepto de Unidad de Trabajo

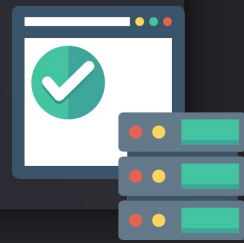
01 Manejo de Temporary Storage

La **TEMPORARY STORAGE o TS** es un recurso que posibilita el pasaje de información entre transacciones. Físicamente se puede almacenar en la Memoria Principal o en Memoria Auxiliar, según los requerimientos del programa.

Puede ser utilizada, por ejemplo, para los siguientes objetivos:

1. Al efectuar la muestra de la selección de varias filas de una tabla o de registros que resulta demasiado grande como para utilizar solo una pantalla para lograrlo. Con la utilización de las PFs podemos acceder a diferentes visiones de la información (Paginado con **TS**).
2. Para pasarle los registros a imprimir a una transacción Asincrónica.
3. Para obtener una lista de registros al final de una cadena de transacciones donde cada una genera uno o más ítems de la **TS** y la última procesa todo el conjunto.

Cuando el programa determina que la **TS** será asignada en Memoria Auxiliar, los Ítems generados se grabarán en un único archivo **VSAM** para cada **CICS** y que fue definido en el stream de arranque del CICS.
Los grupos de registros de **TS** se denominan **COLAS**. Las colas no se definen en ninguna tabla de **CICS,** ya que son definidas dentro de cada programa con indicación de su nombre (hasta 8 caracteres) y grabando uno o más registros utilizando ese nombre.

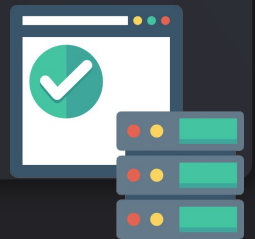


01 Manejo de Temporary Storage

La **COLA** una vez que fue creada, permanecerá hasta que sea borrada por alguna transacción. Cualquier programa que conozca el nombre de la cola podrá acceder a sus datos en forma Secuencial o Random.

Para acceder en forma Random, se debe especificar el nombre de la **COLA** y el número de registro al que se desea acceder **(ITEM)**. Por cada registro que se grabe en la **COLA**, el **ÍTEM** se incrementará automáticamente en 1.

Al asignar el nombre de la cola, es importante considerar que este debe identificarse unívocamente. La forma más tradicional es utilizar los primeros 4 caracteres de los ocho del nombre con el nombre de la transacción que genera la cola y los cuatro restantes con el nombre de la terminal donde se está ejecutando la transacción.



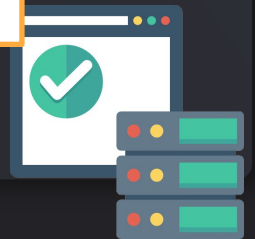
01

Manejo de TS - WRITEQ

El comando **WRITEQ** es usado para grabar nuevos registros o actualizar registros existentes en la Temporary Storage. Si la cola especificada en el comando **WRITEQ TS** ya existe, agrega nuevos registros al final de la cola, caso contrario crea la cola y graba el primer registro.

Si se especifica **ITEM**, el **CICS** retornará el número de ítem del nuevo registro en el área especificada. El número del nuevo registro será el número de ítem del registro anterior más 1. Si **ÍTEM** no se especifica, el número de ítem no será retornado al programa.

REWRITE es usado para que un registro existente en la cola sea actualizado. Si **REWRITE** es especificado, **ÍTEM** también debe ser especificado, colocando el número de ítem que se quiera actualizar. Esto causará que el **WRITEQ TS** actualice el registro de Temporary Storage cuyo número de ítem coincida con el especificado en **ITEM**. No es obligatorio leer la entrada de la cola antes de actualizar, pero es una buena práctica de programación hacerlo.

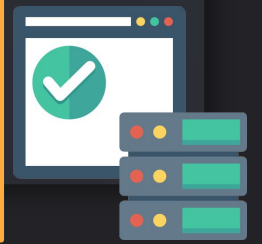


Manejo de TS - READQ

El comando **READQ** permite recuperar un item de **TS.**

```
EXEC CICS READQ TS
        QUEUE    ('name')
        { SET (pointer-ref) : INTO   (data-area) }
        LENGTH (data-value)
        [ ITEM (data-value)  :  NEXT ]
        [ NUMITEMS ]
END-EXEC.
```

- **LENGTH** identifica la longitud del registro a grabar. Los registros de las **TS** no necesitan tener la misma longitud. Es un campo **PIC S9(4) COMP.** Si la longitud del registro excede este valor, será truncado y la condición de excepción **LENGERR** se activará. Para evitar este inconveniente, es recomendable asignar siempre la longitud máxima de registro antes de efectuar la lectura.
- **ÍTEM** en caso de acceso **RANDOM,** especifica el número de registro a leer. Si el registro requerido no existe, la condición de excepción **ITEMERR** se activa. Si la Cola no existe, se activa entonces **QIDERR.**
- **NEXT** específica que se lea el siguiente registro (opción default). Si ya no hay más registros, será activada la condición de excepción **ITEMERR** como consecuencia de haber alcanzado el final de la **TS**.
- **NUMITEMS** es un área de datos a ser actualizada por el **CICS** donde se especifica la cantidad de items total que posee la **COLA.**



Manejo de TS - DELETEQ

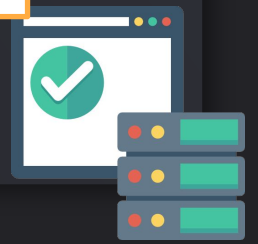
El comando **DELETEQ** permite el borrado de una **COLA.**

```
EXEC CICS DELETEQ TS
          QUEUE    ('name')
END-EXEC.
```

Este comando se puede usar para borrar una cola completa de **TS**. Si la cola no existe, se activa la condición de excepción **QIDERR**

Antes de crear una nueva **COLA**, resulta conveniente ejecutar este comando para asegurarnos que la misma no existe.

IMPORTANTE: No se puede borrar un **ÍTEM** en forma aislada de una **TS**. Si debemos trabajar con bajas de **ÍTEMS,** deberemos habilitar una Flag dentro de nuestro diseño de **COLA** para considerar esta situación.



TS – CONDICIONES DE EXCEPCIÓN

- **ITEMERR** para acceso random, el número de registro especificado no existe en la cola
para acceso secuencial, se alcanzó el fin de la cola

- **LENGERR** el registro leído es más largo que el especificado

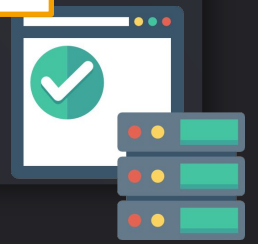
- **NOSPACE** el archivo VSAM en el cual se graban los registros de Temporary Storage carece de espacio

- **IOERR** error de Entrada/Salida

- I**NVREQ** la longitud de los datos a grabar es cero o excede la capacidad especificada para el archivo VSAM

- **QIDERR** no existe la cola especificada

La acción asumida por el CICS en todas estas condiciones de excepción es terminar anormalmente la tarea excepto en el caso de NOSPACE, donde la tarea se suspende hasta que se disponga de espacio.



Mapa Conceptual

01
TEMPORARY STORAGE

02
TRANSIENT DATA

03
UNIDAD DE TRABAJO

02 MANEJO DE TRANSIENT DATA

En aplicaciones batch, podemos especificar un archivo temporal que será borrado al final del job, para pasar datos entre pasos del job.

A diferencia de las colas de temporary storage que pueden ser accedidas en modo random o secuencial tantas veces como se desee, en una cola de **TRANSIENT DATA** los registros son tomados de la cola en el mismo orden en que ellos fueron grabados. No pueden ser actualizados. Si una tarea lee un registro de una cola, el registro se torna inaccesible. Ni esa tarea ni ninguna otra podrá leerlo nuevamente.

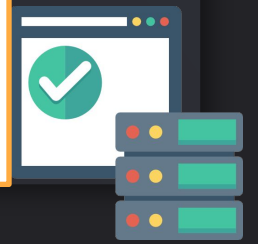
Una cola de **TRANSIENT DATA** debe ser definida en la Tabla de Control de Destinos (Destination Control Table).Esta tabla indica qué tipo de cola transient data será usada: Intrapartition Destination o Extrapartition Destination.

Intrapartition Destination

La utilización de esta cola solo estará disponible dentro del ambiente de **CICS** en la que está definida. Esta cola soporta la función de **TRIGGER**, es decir que por ejemplo cuando supere una cierta cantidad de Ítems se disparará automáticamente la transacción indicada en la **DCT.**

Extrapartition Destination

La utilización de esta cola estará disponible dentro y fuera del ambiente de **CICS** en la que está definida. Puede ser de **INPUT** o de **OUTPUT,** pero no ambas cosas. No permite la función de **TRIGGER.**

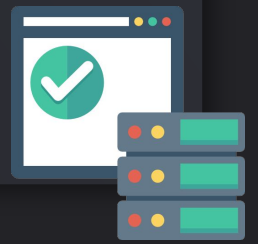


MANEJO DE TD - WRITEQ

El comando **WRITEQ** permite la generar un nuevo item de **TD**.

```
EXEC CICS WRITEQ TD
        QUEUE    ('name')
        FROM     (data-area)
        [ LENGTH (data-value) ]
END-EXEC.
```

- **QUEUE** especifica el nombre de la cola donde se generará el registro nuevo.
- **FROM** identifica el área de trabajo en la que se encuentra el registro a grabar.
- **LENGTH** identifica la longitud del registro a grabar. No debe especificarse si la **TD** es una Extrapartition Transient Data con registros de longitud fija.



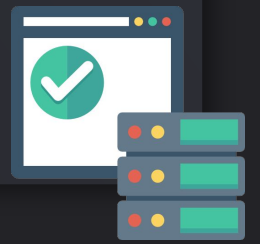
MANEJO DE TD - READQ

El comando **READQ** permite recuperar un ítem de **TD.**

```
EXEC CICS READQ TD
        QUEUE    ('name')
        INTO   (data-area)
        LENGTH (data-value)
END-EXEC.
```

Recordemos que los ítems de las **TD** se acceden en forma secuencial, por lo que no es necesario indicar el **ÍTEM** de **TD** a recuperar.

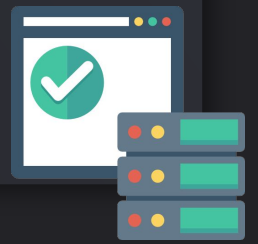
Una vez recuperado el registro, es eliminado automáticamente de la **TD**.



02

TD - CONDICIONES DE EXCEPCIÓN

- **QZERO** la **TD** que se está leyendo está vacía.
- **LENGERR** el registro leído es más largo que el especificado
- **NOSPACE** el archivo **VSAM** de Intrapartition en el cual se graban los registros de **TD** carece de espacio para grabar otro registro.
- **IOERR** error de Entrada/Salida
- **NOTOPEN** El archivo **VSAM** de destino no está abierto en el **CICS**.
- **QBUSY** otra tarea está utilizando el recurso de destino de Intrapartition cuando se ejecuta un comando **READQ.**
- **QIDERR** no existe la cola especificada.



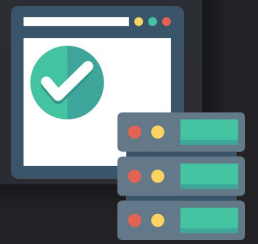
02

MANEJO DE TD - DELETEQ

El comando **DELETEQ** permite el borrado de una **COLA TRANSIENT DATA.**

```
EXEC CICS DELETEQ TD
          QUEUE     ('name')
END-EXEC.
```

IMPORTANTE: Si la **COLA** es una Intrapartition Destination, es posible borrar el contenido de la cola entera con el comando **DELETEQ**. Todo el espacio asignado será liberado. Sin embargo, cualquier transacción CICS puede grabar nuevos registros en la **COLA** que podrán ser leídos por la misma transacción o por cualquier otra.



Mapa Conceptual

01
TEMPORARY STORAGE

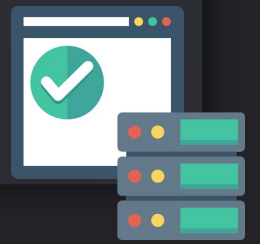
02
TRANSIENT DATA

03
UNIDAD DE TRABAJO

03 CONCEPTO DE UNIDAD DE TRABAJO

UNIDAD LÓGICA DE TRABAJO

- CONCEPTOS
- SYNCPOINT
- SYNCPOINT ROLLBACK
- ABEND



03 CONCEPTO DE UNIDAD DE TRABAJO

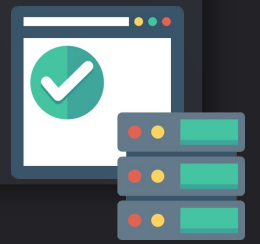
El **CICS** posee una facilidad que es el **DINAMIC TRANSACTION BACKOUT** y consiste en restaurar automáticamente todos los recursos utilizados por una transacción que cancela por condiciones de excepción no contempladas dentro del programa al estado del último **SYNCPOINT** ejecutado, o al estado que tenían antes del inicio de la transacción que cancela si no se ejecutó ningún **SYNCPOINT** dentro del programa.

Para aprovechar al máximo esta facilidad, es necesario conocer qué es una **UNIDAD LÓGICA** de **TRABAJO**, donde comienza y dónde termina.

Una **UNIDAD LÓGICA DE TRABAJO** es el agrupamiento de todas las tareas de actualización de recursos que deben ser ejecutadas por una o varias transacciones, y donde todas deben tener éxito en su ejecución. Si alguna de ellas falla, debemos considerar como de ejecución inválida a todas la transacciones intervinientes, aún a aquellas transacciones ejecutadas exitosamente, porque el resultado general es inválido por la falla en una de sus partes.

La UNIDAD LÓGICA DE TRABAJO comienza con la ejecución de la primera transacción de actualización de lo que nosotros, en nuestro diseño de la aplicación, consideramos como una **UNIDAD LÓGICA DE TRABAJO**, y finaliza cuando:

A. Termina la Transacción de mayor nivel
B. Al momento de la ejecución del comando SYNCPOINT



03 CONCEPTO DE UNIDAD DE TRABAJO

El **CICS**, por su parte, efectúa un **SYNCPOINT** (o punto de sincronismo) automático si la transacción principal termina exitosamente, es decir sin ninguna cancelación por condiciones de excepción.

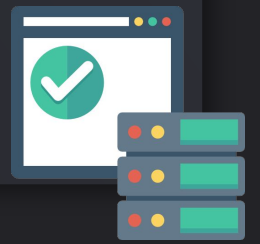
Un ejemplo de **UNIDAD LÓGICA de TRABAJO** podría ser que un cliente efectúa una transferencia de dinero desde su cuenta de ahorro a su cuenta corriente. Esto se puede dividir en tres procesos

1. Efectuar el débito en la caja de ahorro
2. Efectuar el crédito en la cuenta corriente
3. Registrar el movimiento de transferencia

El conjunto de las tres transacciones podríamos considerarlo como nuestra **UNIDAD LÓGICA de TRABAJO**, ya que si alguna de ellas no se ejecuta en forma satisfactoria, toda la operación debe ser considerada como no efectuada.

Solamente pensemos qué pasaría si la transacción 1) finaliza normalmente, la transacción 2) finaliza normalmente y la transacción 3) falla en la registración. Tendríamos un problema en nuestros registros.

En este caso, deberíamos efectuar una vuelta atrás o **BACKOUT** de lo hecho por las transacciones 1) y 2) para invalidar toda la transacción en su conjunto y dar cuenta al usuario operador del problema encontrado al momento de ejecutar la transacción 3).



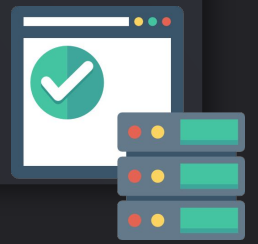
ABEND

El comando **ABEND** además de restaurar todos los recursos actualizados al estado inicial de nuestra **UNIDAD LÓGICA DE CONTROL**, permite efectuar un vuelco de memoria para su posterior análisis.

```
EXEC CICS   ABEND
        [ ABCODE (name) ]
        [ CANCEL ]
END-EXEC.
```

- **ABCODE** producirá un vuelco de memoria que será almacenado en el archivo especificado por 'name'
- **CANCEL** produce la suspensión de cualquier rutina de **HANDLE ABEND** vigente (para no entrar en loop).

Si nuestra aplicación utiliza recursos como el **DL/I o DB2**, el comando **ABEND** también restaura todos los cambios efectuados en ellos.

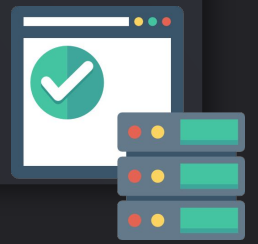


03

Transacciones propias de CICS

Transactions (1 of 2)

Function	CICS-supplied Transaction
Sign On	CESN
Sign Off	CESF
Monitor and Control CICS Resources	CEMT
Message Switching	CMSG
Resource Definition Online (RDO)	CEDA
	CEDB
	CEDC
CICS DB2 Interface	DSNC or CEMT

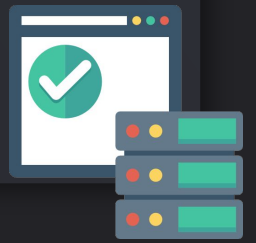


03

Transacciones propias de CICS

Transactions (2 of 2)

Function	CICS-supplied Transaction
Processing and Debugging Programs	CMAC
	CECI
	CEBR
	CEDF
	CEDX
Database Control Inquiry	CDBI
Database Control Interface	CDBM
Database Control Menu	CDBC



¡Muchas gracias!