Bases de Datos Relacionales

PROCEDIMIENTOS ALMACENADOS

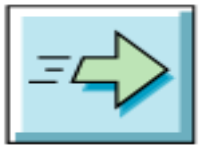
Preparó: ***Ismael Castañeda Fuentes***

Fuentes: **Manuales Sybase**
Manuales SQL Server
Manuales Oracle

Procedimientos almacenados

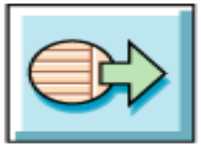
- Un procedimiento almacenado es un conjunto de sentencias SQL y de control de flujo
- Beneficios de los procedimientos almacenados:
  - Simplifican la ejecución de tareas repetitivas
  - Corren más rápido que las mismas instrucciones ejecutadas en forma interactiva
  - Reducen el tráfico a través de la red
  - Pueden capturar errores antes que ellos puedan entrar a la base de datos
  - Establece consistencia porque ejecuta las tareas de la misma forma
  - Permite el desarrollo modular de aplicaciones
  - Ayuda a proveer seguridad
  - Puede forzar reglas y defaults complejos de los negocios

Procedimientos - Tipos

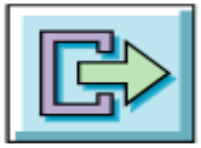


- Procedimientos almacenados definidos por el usuario
  - Son procedimientos definidos por el usuario que se deben llamar explícitamente

- Triggers
  - Son procedimientos definidos por el usuario que se ejecutan automáticamente cuando se modifica un dato en una tabla

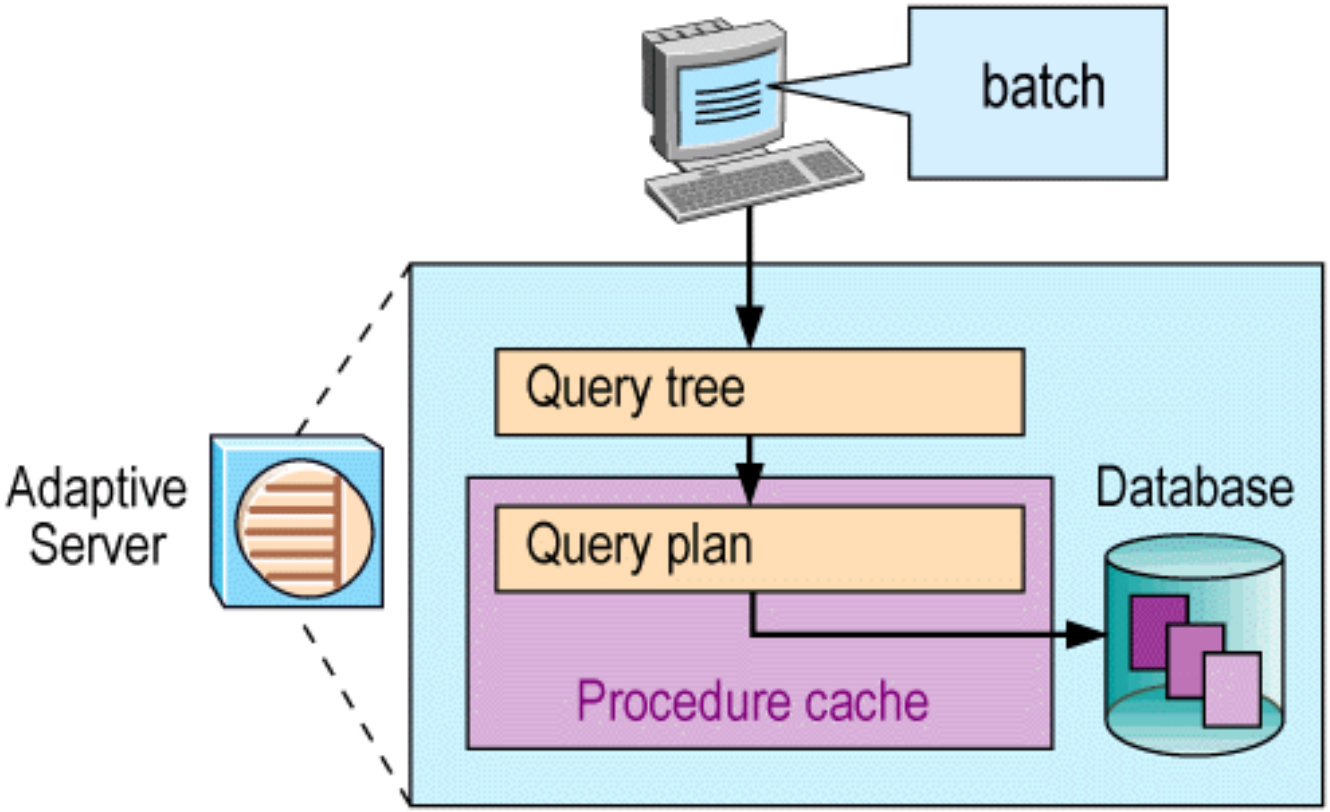


- Procedimientos del sistema
  - Procedimientos suministrados por el sistema

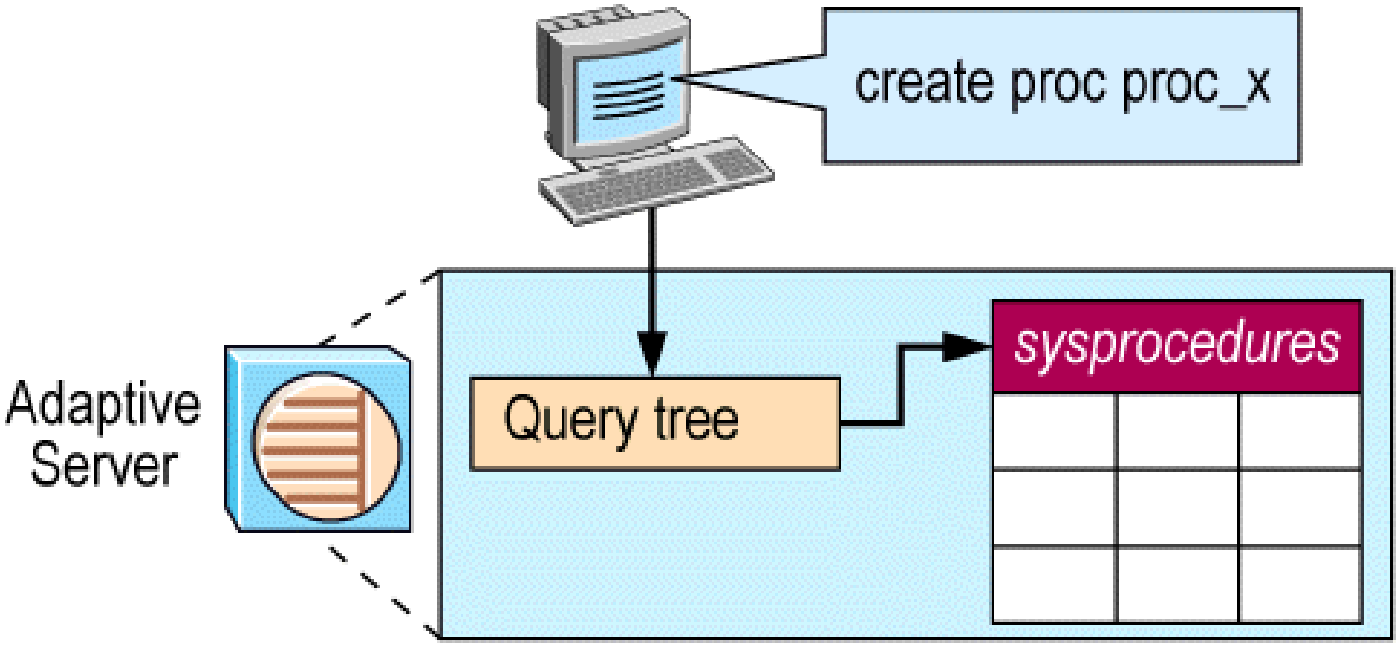


- Procedimientos extendidos
  - Procedimientos que hacen llamadas al sistema operativo y ejecutan tareas a ese nivel

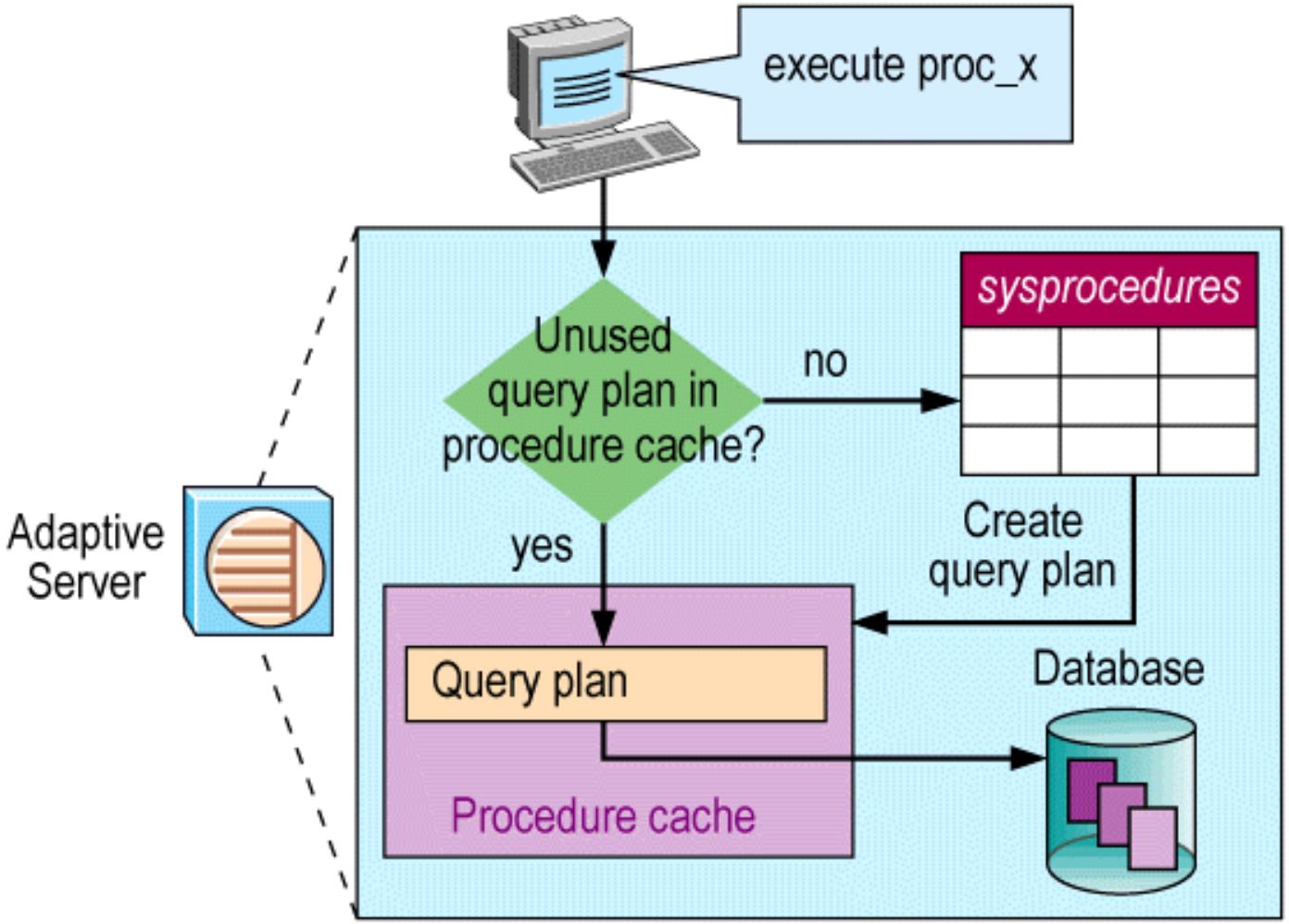
Interactive Execution



Procedimientos - Creación



Procedimientos - Ejecución



Procedimientos - Ventajas en el rendimiento

Un procedimiento almacenado se ejecuta más rápido que un batch porque:

- El procedimiento almacenado ya ha sido analizado
- Ya se han resuelto las referencias a los objetos referenciados en el procedimiento almacenado
- No se necesita construir el árbol de búsqueda, él usa el que se hace en el momento de compilarlo
- No se necesita crear un plan de búsqueda, porque ya el procedimiento tiene uno

Procedimientos - Crear y borrar

- Sintaxis simplificada para create:

```
create proc procedure_name
    as
        statements
    return
```

- Sintaxis simplificada para drop:

```
drop proc procedure_name
```

Procedimientos - Ejecución

- Sintaxis simplificada:
  [**exec** | **execute**] *procedure_name*

Procedimientos - Variables

- Los procedimientos almacenados pueden crear y usar variables locales
  - Las variables sólo existen mientras exista el procedimiento
  - Las variables no las puede usar otro proceso

Procedimientos - Sentencias válidas e inválidas

- Un procedimiento almacenado puede:
  - Seleccionar y modificar datos
  - Crear tablas temporales y permanentes
  - Llamar otros procedimientos almacenados
  - Referenciar objetos de bases de datos

- Un procedimiento almacenado no puede ejecutar:
  - **use *database***
  - **create view**
  - **create default**
  - **create rule**
  - **create procedure**
  - **create trigger**

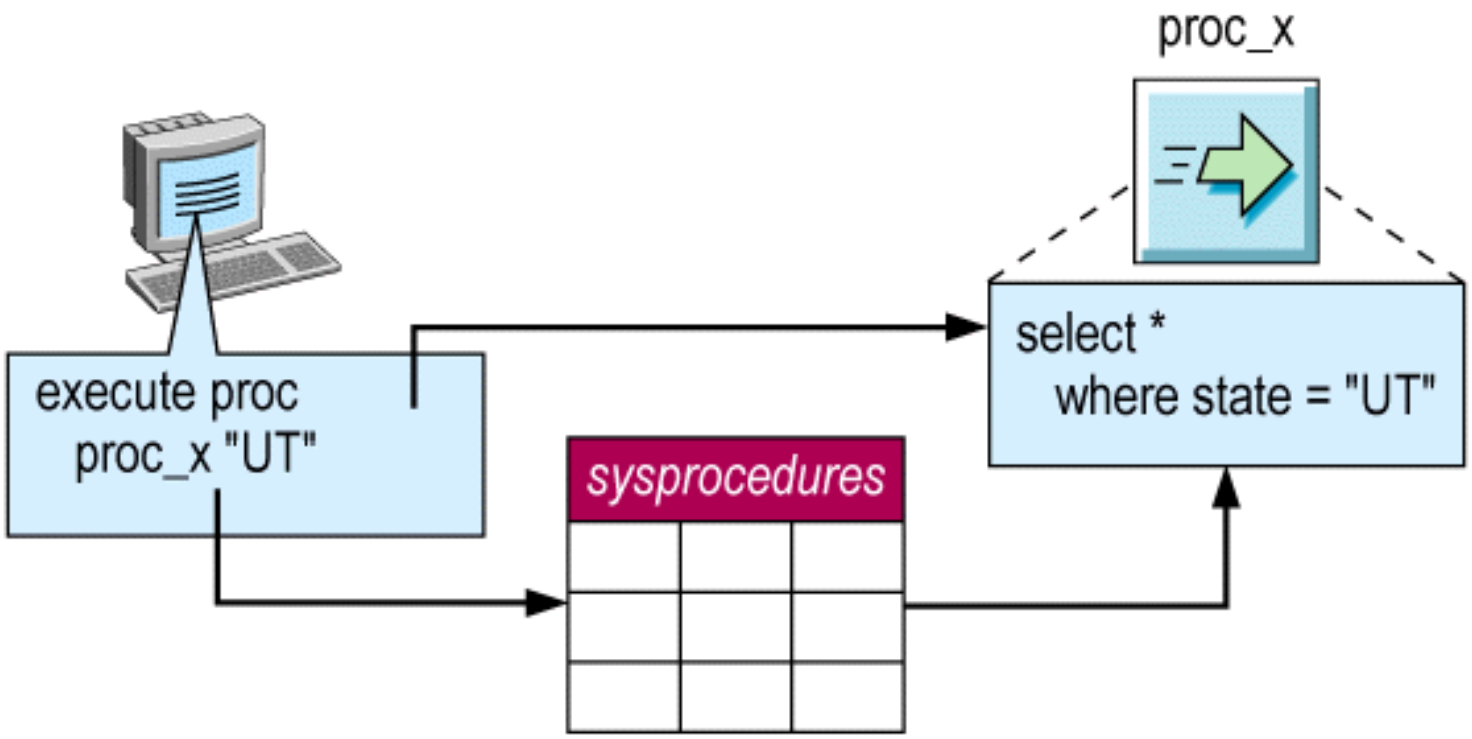
Procedimientos – Permisos

- Para permitir que otros usen un procedimiento almacenado, el propietario debe dar los respectivos permisos

- Sintaxis simplificada:
  grant execute
  on *procedure_name*
  to *user_list*

Procedimientos – Parámetros de entrada

- Un Parámetro de entrada es una variable local del procedimiento almacenado que puede recibir un valor de una sentencia **exec procedure *lista_de_parámetros***



Procedimientos – Parámetros de entrada

- Sintaxis simplificada:

create procedure *procedure_name*

(parameter_name datatype default_value

[*, parameter_name datatype default_value...*])

as

statements

return

Procedimientos – Paso de parámetros

- Dos métodos para pasar valores a parámetros:
  - Paso de parámetros por posición
  - Paso de parámetros por nombre

Procedimientos – Paso de parámetros por posición

- Sintaxis para paso por posición:
  [exec | execute] *procedure_name value* [, *value*...]
- Los parámetros se deben pasar en el mismo orden en que ellos aparecen en la sentencia **create procedure**
- Como este método es más propenso a errores, se aconseja el paso por nombre

Procedimientos – Paso de parámetros por nombre

- Sintaxis para paso por nombre:
  [**exec** | **execute**] **procedure** *procedure_name*
  parameter_name = *value* [, *parameter_name* = *value*]

- Los nombres de los parámetros en la sentencia **exec** deben concordar con los nombres de los parámetros usados en la sentencia **create procedure**

- Los parámetros pueden pasar en cualquier orden

Procedimientos – Valores por default

- Se puede asignar un valor por default a un parámetro cuando él no se indica en la sentencia **exec**

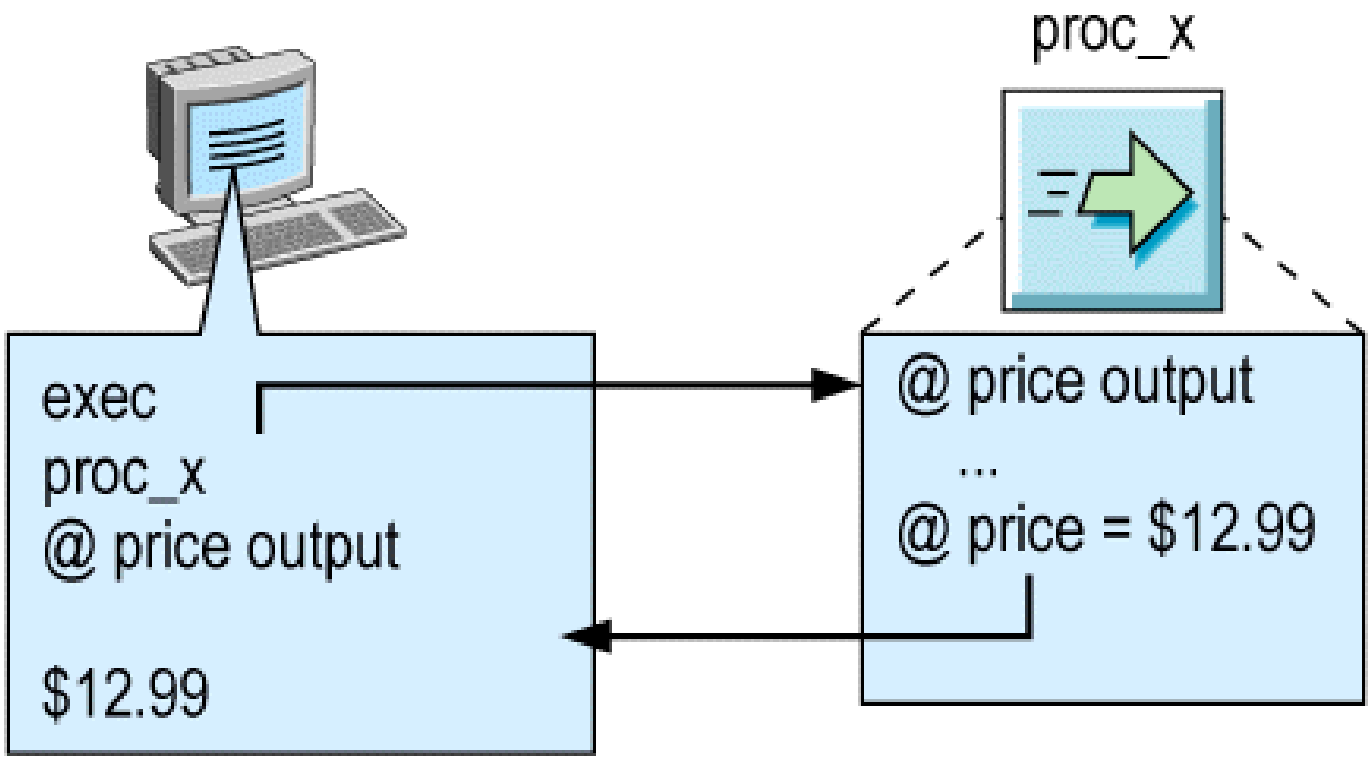
. . .

Procedimientos – Errores típicos en parámetros de entrada

- Los valores que se pasan no tienen el mismo tipo de datos que los parámetros definidos
- En la misma sentencia, se pasa un parámetro por posición después de haber pasado un parámetro por nombre
  - Aunque no es recomendado, es posible mezclar los dos métodos para pasar valores, sin embargo, después de pasar un valor a un parámetro por nombre, todos los restantes de deben pasar por nombre
- Olvido de uno o más parámetros
  - El olvido de uno o más valores para los parámetros, hace que se usen los valores por default
- Los valores para los parámetros se pasan en un orden errado

Procedimientos – Retorno de valores

- Un parámetro output es una variable local en un procedimiento almacenado que se puede enviar a una Sentencia **exec procedure**



Procedimientos – Crear parámetros que retornan valores

- Sintaxis simplificada:

```
create procedure procedure_name
        (parameter_name datatype output
        [, parameter_name datatype output...] )
as
        statements
return
```

Procedimientos – Usar parámetros que retornan valores

- Sintaxis simplificada:
  [**exec | execute**] *procedure_name variable* **output**

- Los valores que retornan los parámetros se pasan automáticamente al conjunto respuesta

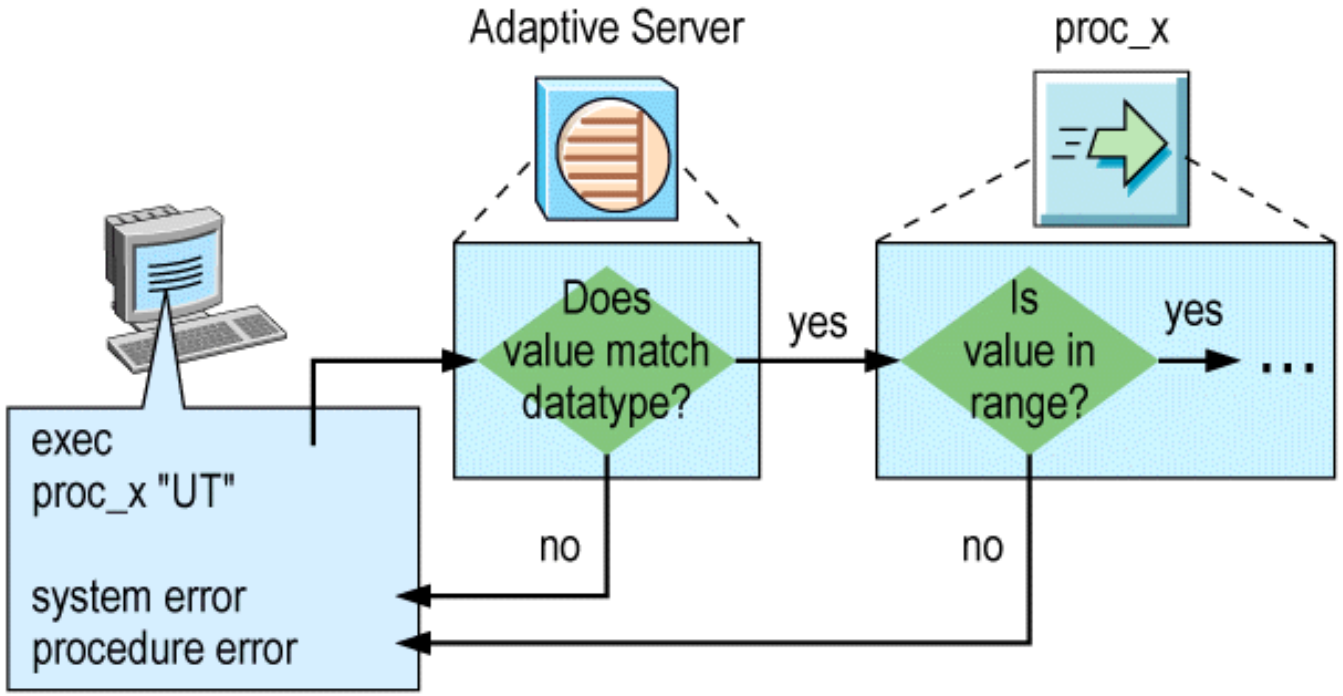
- El retorno de valores se pueden pasar por nombre o por posición
  - Se recomienda el paso por nombre

Legibilidad

- Para hacer un código más legible:
  - Utilizar comentarios
  - Utilizar indentación
  - Usar espacios en blanco y dejar el código organizado en columnas
  - Declarar e iniciar las variables en un bloque
- Establecer un conjunto de buenas prácticas para legibilidad

Procedimientos – Chequeo de valores

- El servidor chequea que los parámetros concuerden en tipo de datos
- Los procedimientos almacenados verifican que los valores pasados se encuentren dentro del dominio establecido



Procedimientos – mensajes de error

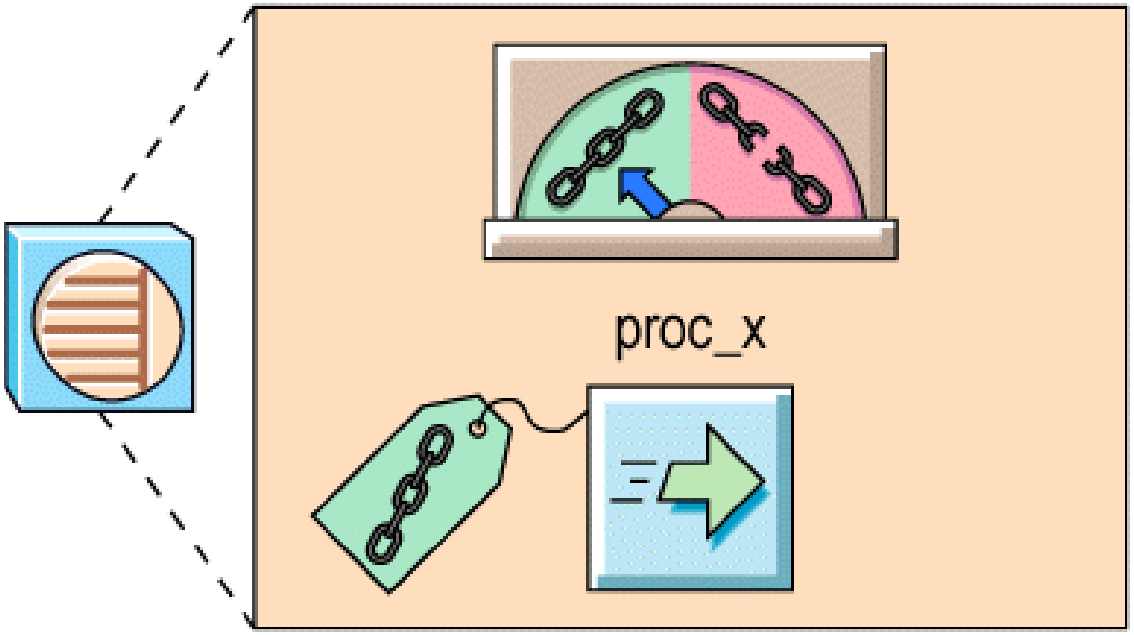
- Si un procedimiento almacenado requiere valores para los parámetros, el usuario debe incluir un tratamiento de esos errores y un manejo de mensajes de error

- Ejemplo:

```
create proc proc_cutoff
      (@title_id     char(6) = NULL,
       @max_price    money           = NULL)
as
      if @title_id is NULL or @max_price is NULL
      begin
              raiserror 20001
              "Execution for this procedure is:
              exec proc_cutoff title_id, max_price"
              return
      end
      ...
return
```

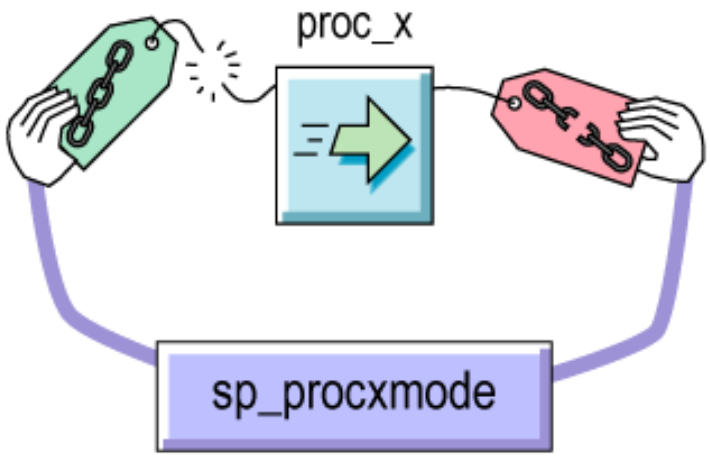
Procedimientos – Rótulos de modo de transacción

- Los procedimientos almacenados se rotulan con el modo de transacción con el cual fueron creados
  - No se puede ejecutar una transacción en un modo diferente al del rótulo



Procedimientos – sp_procxmode

- **sp_procxmode** permite ver y cambiar el modo de transacción de un procedimiento

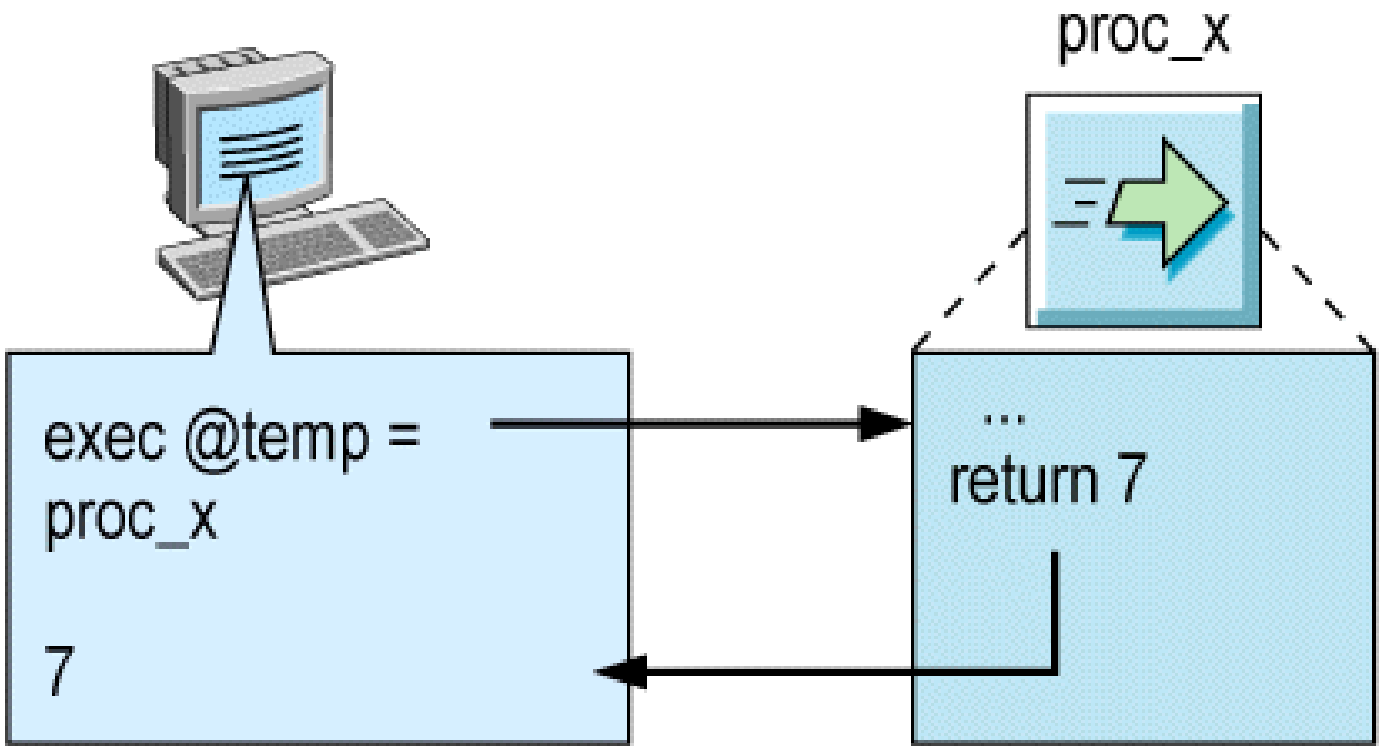


- Sintaxis:
  sp_procxmode [*procedure_name* [, {**chained** | **unchained** | **anymode**}]]

- Ejemplo:
  ```
  sp_procxmode proc_update_titles, unchained
  ```

Procedimientos – Retorno de valores de estado

- A return status es un valor que indica cuándo o no un procedimiento se ha ejecutado totalmente



Procedimientos – Retorno de valores de estado

Values	Meaning
Greater than 0	No predefined meaning; available for user-detected errors
0	Successful completetion
-1 through –99	System-detected error
Less than –99	No predefined meaning; available for user-detected erros

Procedimientos – Retorno de valores de estado

- Sintaxis simplificada:
  create proc *procedure_name*
  as
  statements
  return [*return_status*]

Procedimientos – Captura del valor del estado de retorno

- Los valores de retorno se deben capturar en variables
- Sintaxis simplificada:
  [**exec | execute**] *variable = procedure_name parameter_list*

Procedimientos – Rollbacks no intencionales

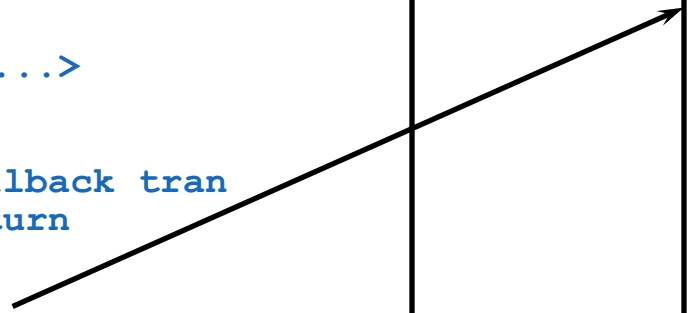
Un **rollback** no intencional es un **rollback** anidado que sin intensión deshace el trabajo en transacciones externas

Proc1(Outermost tran)

```
begin tran
<statements...>
if <error>
   begin
         rollback tran
         return
   end
exec proc2
if <error>
   begin
         rollback tran
         return
   end
<statements...>
commit tran
return
```

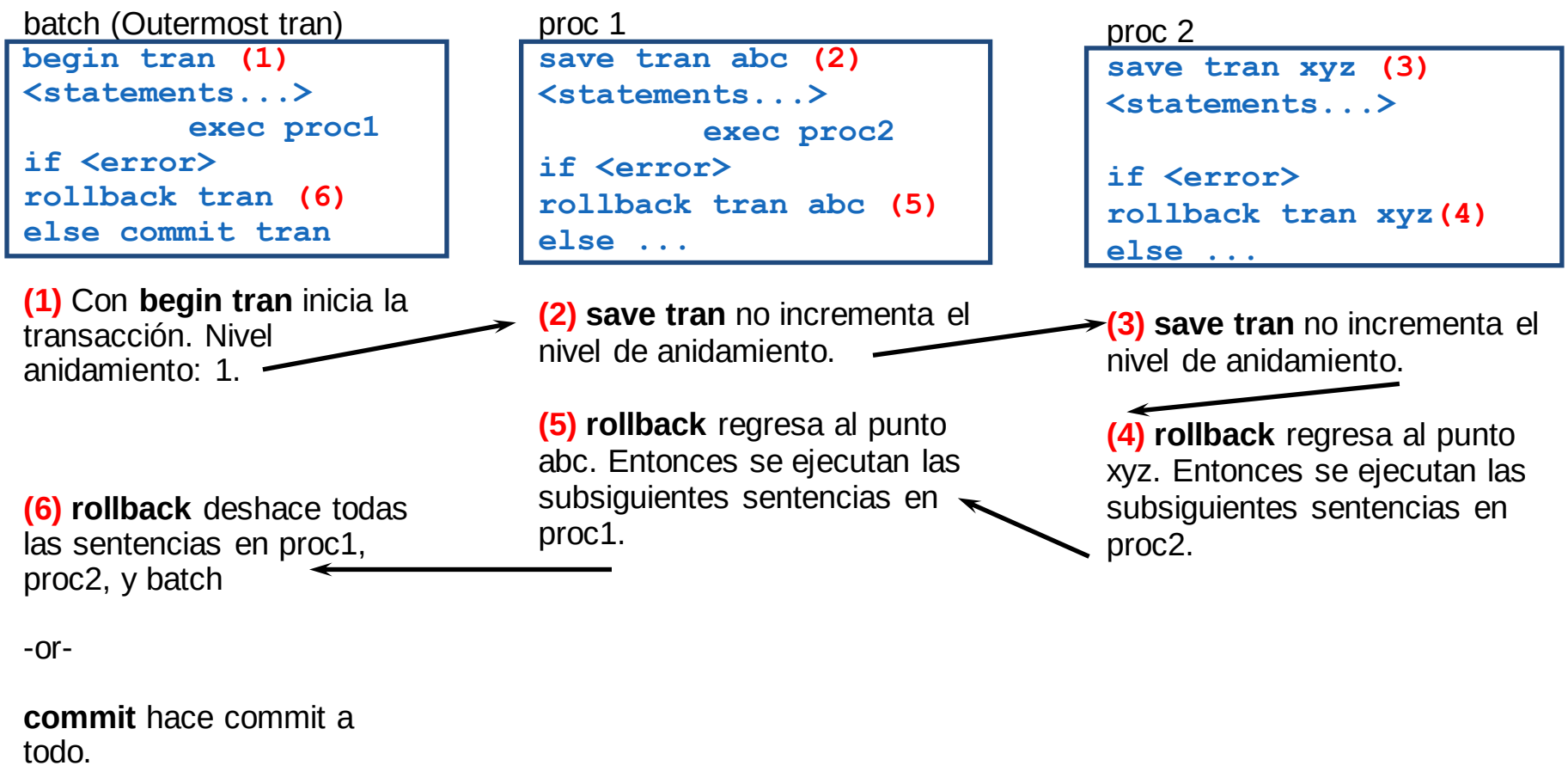
Proc2(Nested proc)

```
begin tran
<statements...>
if <error>
   begin
         rollback tran
         return
   end
<statements...>
commit tran
return
```



Procedimientos – Savepoints y Rollbacks anidados

Usar savepoints para evitar rollbacks no intencionales



Procedimientos – Límite de anidamiento

- Los procedimientos almacenados pueden llamar otros procedimientos almacenados
  - El máximo nivel de anidamiento es 16
  - La variable @@*nestlevel* contiene el nivel de anidamiento actual
- Si se excede el nivel máximo:
  - Se abortan los procedimientos pendientes
  - El servidor retorna un error

Planes de búsqueda

- Un plan de búsqueda es un conjunto ordenado de estapas que se requieren para acceder los datos, incluyendo información sobre:
  - Si usar o no un índice
  - El índice a usar
  - El orden en el cual las tablas se deben encadenar
- Los planes de búsqueda son creados por el optimizador de búsquedas
  - El optimizador de búsquedas usa información acerca de los objetos de base de datos para producir el plan
- Los planes de búsqueda creados para los procedimientos, se reutilizan
  - Cuando se ejecuta un procedimiento almacenado, el servidor chequea el caché del procedimiento para un plan no usado
  - Si hay un plan de búsqueda no utilizado, el servidor lo usa
  - Si no hay un plan de búsqueda no utilizado, el servidor genera uno nuevo del árbol de búsqueda en *sysprocedures*

Planes de búsqueda sub-óptimos

- El plan de búsqueda creado para la una ejecución de un procedimiento almacenado puede que no sea el plan de búsqueda óptimo para la siguiente ejecución del procedimiento almacenado
  - Las dos ejecuciones pueden usar parámetros de entrada muy diferentes
  - Se pueden haber añadido nuevos índices entre las dos ejecuciones
  - El tamaño de las tablas accesadas pueden haber cambiado significativamente entre las dos ejecuciones
- Hay tres formas para forzar al servidor a generar un nuevo plan de búsqueda
  - Usar **with recompile** en el procedimiento
  - Usar **with recompile** cuando se ejecute el procedimiento
  - Usar **sp_recompile**

Procedimientos con recompile

- En un procedimiento, usar la opción **with recompile** para forzar al servidor a crear un nuevo plan de búsqueda cada vez que se ejecute el procedimiento

- Sintaxis simplificada:

```
create proc procedure_name
with recompile
as
        statements
return
```

Procedimientos con recompile

- Cuando se ejecute un procedimiento almacenado, usar la opción **with recompile** para forzar al servidor a crear un nuevo plan de búsqueda para esa ejecución del procedimiento
- Esta opción se puede usar cuando se ejecuta cualquier procedimiento almacenado
- Sintaxis simplificada:
  [exec | execute] *procedure_name* with recompile

sp_recompile

- **sp_recompile** hace que cada procedimiento almacenado (y trigger) que utilice la tabla indicada se recompile la siguiente vez que él se ejecute
- Sintaxis:
  sp_recompile *table_name*