PROGRAMACIÓN DE BASE DE DATOS

Triggers.

Procedimientos personalizados.

ING. JAIME LLANOS BARDALES

Introducción

- **Quiz**

Poll Everywhere

Agenda

Disparadores y bucles

Objetivo: Creación y gestión de disparadores para generar tablas de auditoría.

- Disparadores en SQL.

Triggers

Los Triggers constituyen un tipo especial de procedimiento almacenado

- Se ejecuta automáticamente cuando se produce un evento de manipulación de datos (INSERT, UPDATE o DELETE).
- Los desencadenadores DML pueden usarse para aplicar reglas de negocios y la integridad de datos, consultar otras tablas e incluir instrucciones Transact-SQL complejas.

```
-- SQL Server Syntax
-- Trigger on an INSERT, UPDATE, or DELETE statement to a table or view (DML Trigger)

CREATE [ OR ALTER ] TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }

<dml_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]

<method_specifier> ::=
    assembly_name.class_name.method_name
```

Triggers

- Los Triggers constituyen un tipo especial de procedimiento almacenado

- Los triggers definen dos tablas especiales que contienen la información de la operación: ***inserted*** y ***deleted***.
- Ambas se crean a partir de la estructura de la tabla afectada y contienen los registros afectados en la operación.

  - ***inserted*** contiene los registros con nuevos valores según sea la operación: INSERT (nuevos
  - registros) y UPDATE (nuevo valor para registros actualizados).

  - ***deleted***, contiene los registros con los viejos valores según sea la operación: DELETE (registros
  - borrados) y UPDATE (valor anterior para los registros actualizados).

- ***AFTER INSERT*** dispondremos de la tabla inserted
- ***AFTER DELETE*** solamente tendremos la tabla deleted
- ***AFTER UPDATE*** ambas tablas estarán disponibles, pudiendo consultar así de los valores antes y después de
  - actualizarse los registros correspondientes

Triggers

```
-- ------------------------------------------------------------
-
-- Creamos una tabla si no existiese.
-- Representa los datos de expedientes
-- ------------------------------------------------------------
-

IF NOT EXISTS (SELECT * FROM sys.sysobjects WHERE name='expedientes'
AND xtype='U')
BEGIN
    CREATE TABLE expedientes (
      code             VARCHAR(15)  NOT NULL,
      state            VARCHAR(20)  DEFAULT 'INICIO',
      stateChangedDate DATETIME,
      PRIMARY KEY (code)
    );
END;

-- Insertamos algunos expedientes de ejemplo
DELETE FROM expedientes WHERE code IN ('exp1','exp2', 'exp3');
INSERT INTO expedientes (code) VALUES ('exp1');
INSERT INTO expedientes (code) VALUES ('exp2');
INSERT INTO expedientes (code) VALUES ('exp3');

select * from expedientes;
```

```
-- Si no existe la tabla de cambios de estado la
creamos
IF NOT EXISTS (SELECT * FROM sys.sysobjects WHERE
name='expStatusHistory' AND xtype='U')
BEGIN
    CREATE TABLE expStatusHistory (
      id    INT          IDENTITY,
      code  VARCHAR(15) NOT NULL,
      state VARCHAR(20) NOT NULL,
      date  DATETIME    DEFAULT GetDate(),
      PRIMARY KEY  (id)
);
END;
```

Triggers

```
-- Borramos el Trigger si existise
IF OBJECT_ID ('StatusChangeDateTrigger', 'TR') IS NOT NULL
BEGIN
   DROP TRIGGER StatusChangeDateTrigger;
END;

-- Crememos un Trigger sobre la tabla expedientes
CREATE TRIGGER StatusChangeDateTrigger
ON expedientes
 AFTER UPDATE AS
 -- ¿Ha cambiado el estado?
 IF UPDATE(state)
 BEGIN
-- Actualizamos el campo stateChangedDate a la fecha/hora actual
UPDATE expedientes SET stateChangedDate=GetDate() WHERE code=(SELECT code
FROM inserted);

    -- A modo de auditoría, añadimos un registro en la tabla
expStatusHistory
INSERT INTO expStatusHistory  (code, state) (SELECT code, state FROM
deleted WHERE code=deleted.code);

    -- La tabla deleted contiene información sobre los valores ANTIGUOS
mientras que la tabla inserted contiene los NUEVOS valores.
    -- Ambas tablas son virtuales y tienen la misma estructura que la tabla
a la que se asocia el Trigger.
 END;
```

```
UPDATE expedientes SET state='PENDIENTE_COBRO' WHERE code='exp1'

 select * from expedientes;
 SELECT * FROM expStatusHistory;
```

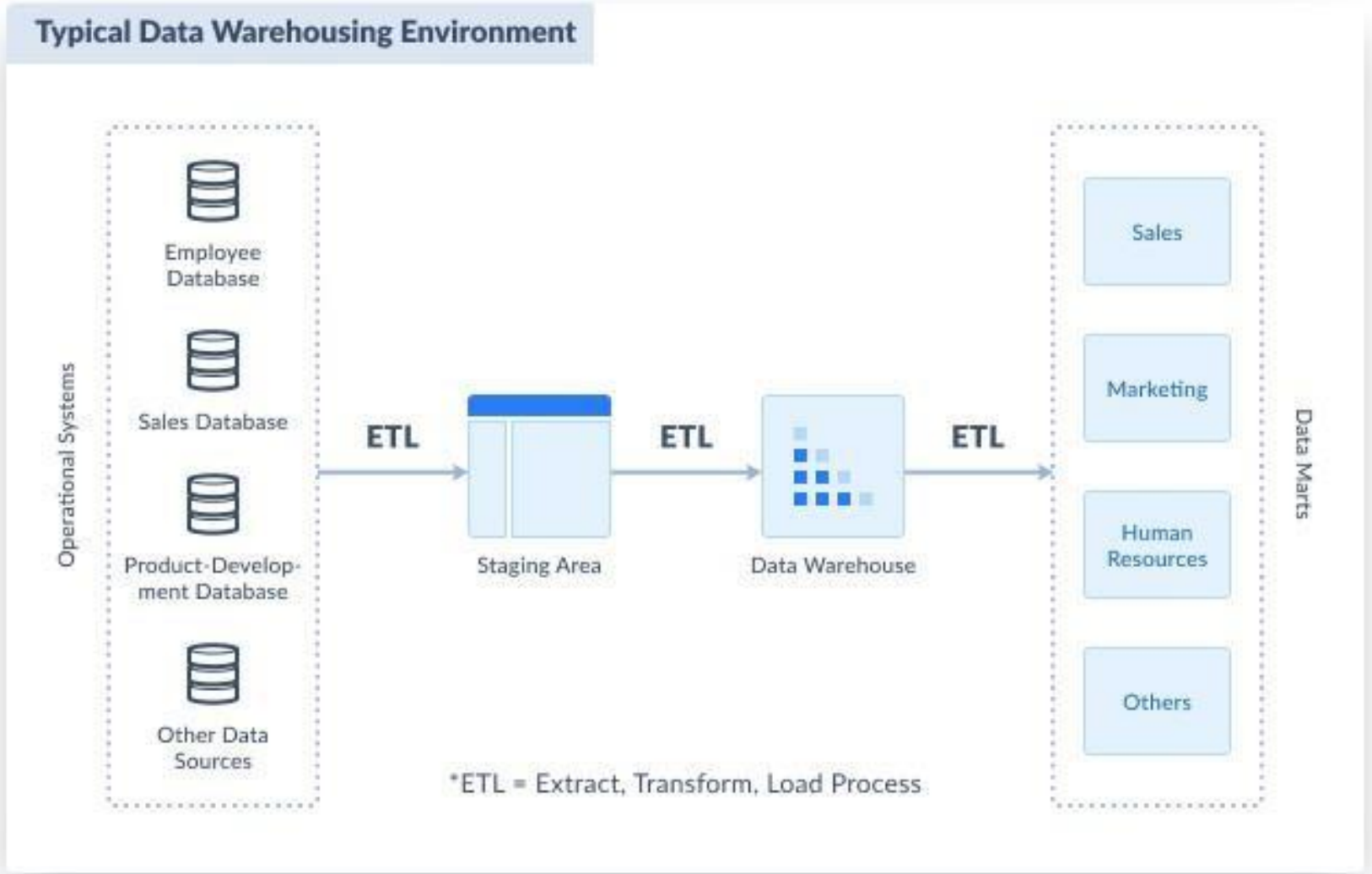
Agenda

PROCEDIMIENTOS Y FUNCIONES PERSONALIZADAS

- Revisión de caso de negocio de carga de diversas fuentes usando SQL Server

Que es un DataWareHouse (DWH)

Un DWH es un repositorio información para el análisis de datos, la inteligencia artificial y el machine learning. Los datos fluyen desde diferentes fuentes de datos, como bases de datos transaccionales. Estos datos se reciben a un corte de los sistemas transaccionales y se acopian. Para encontrar los indicadores que requiere el negocio.



Herramientas de Trabajo

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Online

SQL Online

SQL Server Management Studio

v. 19.1

Microsoft SQL Server

Fin de la sesión