Facultad de Ciencia Y Tecnología

Base de Datos Avanzadas

Bases de Datos Activas

DOCENTES:

- TROSSERO, SEBASTIÁN
- SCHMUCKLER, JORGE

Resumen

- Introducción
- Características
- Paradigma ECA
- Reglas Activas
- Casos de estudio

Introducción – BD Pasivas

Son las bases de datos que conocemos hasta el momento por la materia Bases de datos (estándar SQL-92)

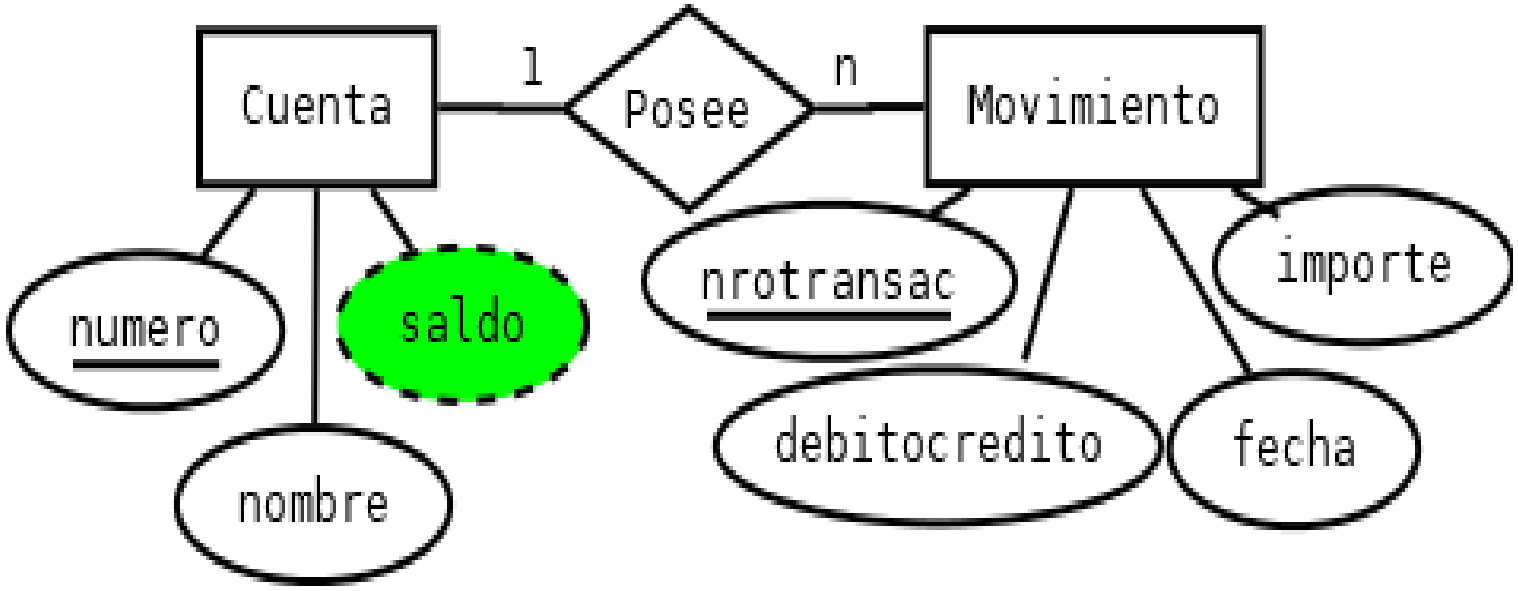
Todas las acciones sobre los datos resultan de invocaciones **explícitas** de los programas de aplicación.

Introducción – Caso de estudio

Supongamos el ejemplo de un Banco “B” que permite realizar una transacción denominada **TRANSFERENCIA DE CUENTAS INTERNAS**, extrayendo un Importe determinado de una cuenta Origen (débito), y acreditándolo en una cuenta Destino (crédito).

- Este débito y crédito se debe registrar en la tabla **MOVIMIENTOS**.
- REGLA DE NEGOCIO: todo **Movimiento** debe ajustar el atributo **Saldo** de la tabla **cuentas**.

Introducción – Caso de estudio



* Si bien “saldo” es un atributo derivado, por la regla de negocio enunciada se definió como atributo almacenado.

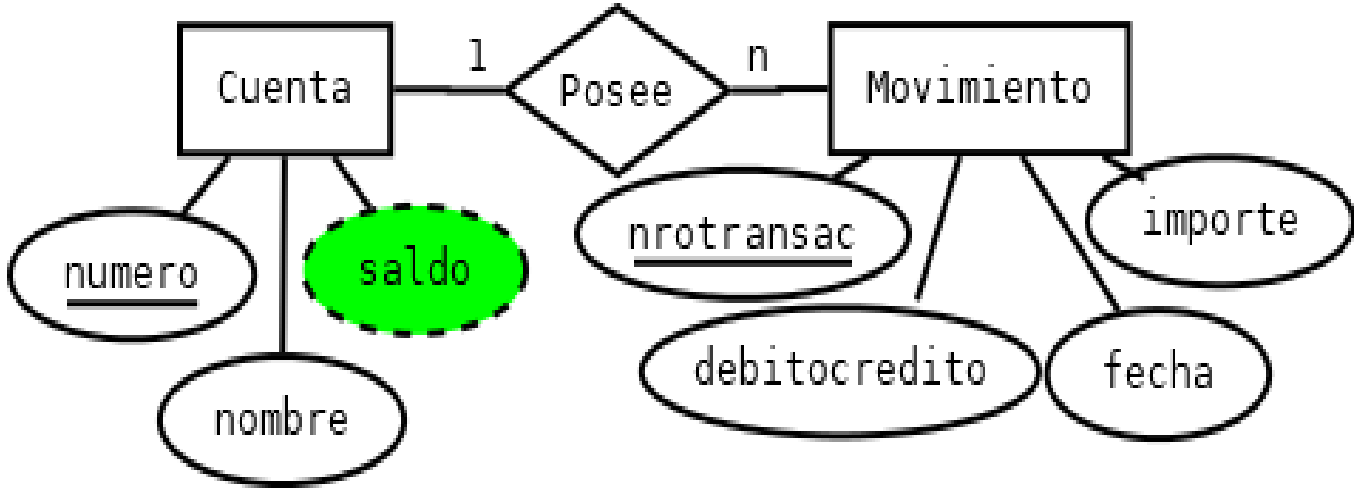
Introducción – Caso de estudio

```
CREATE TABLE cuentas (
  Numero int not null primary key,
  Nombre varchar(50),
  Saldo numeric(15,2)
);

CREATE DOMAIN debitocredito AS
Char(1) check(
  value in ('D','C')
);
```

```
CREATE TABLE movimientos (
  NumeroTransaccion int not null,
  Fecha Date,
  CuentaNumero int,
  DebitoCredito debitocredito,
  Importe Numeric(15,2),
  Foreign key (CuentaNumero)
  References Cuentas(numero)
);
```

Introducción – Caso de estudio

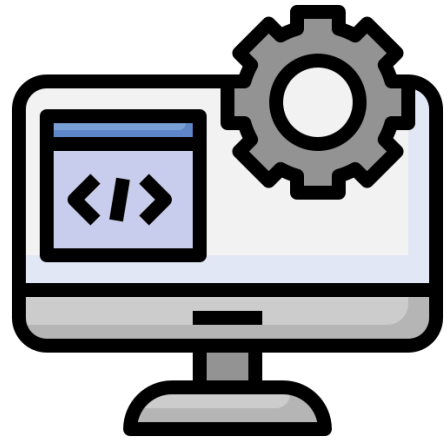


Regla de Negocio

- Si insertamos un movimiento de DebitoCredito con **'D' = débito**, el saldo de la cuenta asociada se **decrementará** en valor correspondiente a importe.
- Si insertamos un movimiento de DebitoCredito con **'C' = crédito**, el saldo de la cuenta asociada se **incrementará** en valor correspondiente a importe.

Introducción – Caso de estudio
Problema

Pipeline de ejecución de una transferencia bancaria entre cuentas:



Software de aplicación

1° INSERT débito en cuenta origen

2° INSERT crédito en cuenta destino

3° UPDATE saldo en cuenta origen

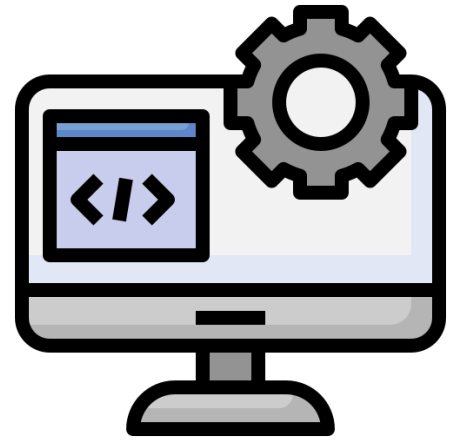
4° UPDATE saldo en cuenta destino

SGBD

Comportamiento completamente **PASIVO**

Introducción – Caso de estudio
Solución

Pipeline de ejecución de una transferencia bancaria entre cuentas:



Software de aplicación

1° INSERT débito en cuenta origen

2° INSERT crédito en cuenta destino

Disparador (Trigger)

SGBD

AUTOMÁTICO

3° UPDATE saldo en cuenta origen

4° UPDATE saldo en cuenta destino

Comportamiento **ACTIVO** realizado por trigger de SGBDA

Resumen

- Introducción
- Características
- Paradigma ECA
- Reglas Activas
- Casos de estudio

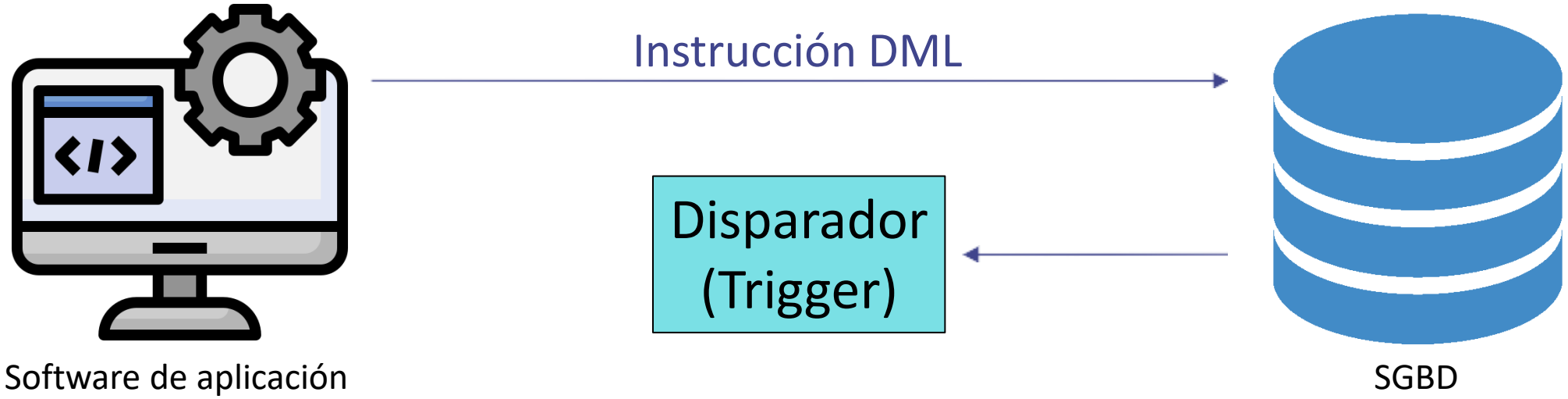
Características

Los SGBD Activos (SGBD**A**):

- Son capaces de detectar situaciones de interés y de actuar en consecuencia.
- Debe ser capaz de monitorear y reaccionar ante eventos de manera oportuna y eficiente.
- Proporcionan mecanismos para definir “**cuando**” y “**qué**” mediante la definición de **Reglas Activas**.
- Opcionalmente permiten la evaluación de una **“condición”** que determina si se ejecuta el **“qué”**
- Deben tener la capacidad de almacenar las Reglas Activas al igual que se almacenan definiciones de Tablas y Vistas

Características

Entonces, el concepto básico de un comportamiento activo queda:



Regla de negocio => Regla Activa

Características

Comportamiento ACTIVO = **CUANDO** + **QUÉ**

En el caso de estudio,

"REGLA DE NEGOCIO: todo **Movimiento** debe ajustar el atributo **Saldo** de la tabla **cuentas**."

CUANDO se inserte un movimiento entonces **se actualiza el saldo** de la cuenta asociada.

Nota: En este caso no se presenta condición, **se ejecuta siempre**.

Características

¿Y qué pasa con la Independencia y la Consistencia?

Características
Independencia y Consistencia

Mediante los SGBDA se logra un **mayor nivel de independencia y consistencia** en sus datos.

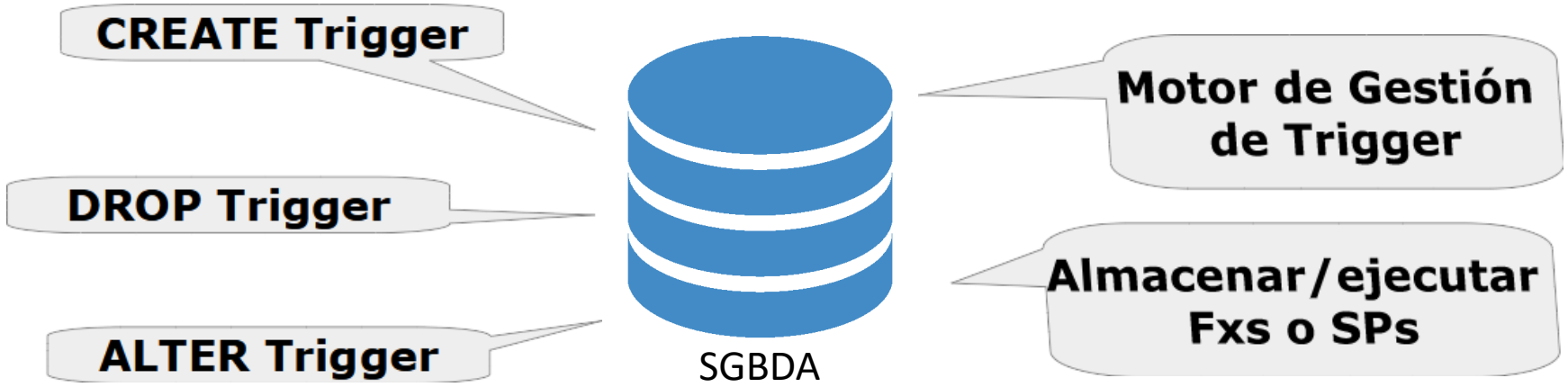
Al estar definidas como parte del esquema las **reglas de integridad** que se deben cumplir, se elimina la dependencia de las aplicaciones para que realicen estos controles.

Además permite que un cambio de comportamiento se puede llevar a cabo en la misma base de datos sin necesidad de modificar las aplicaciones.

Características

En resumen podemos definir:

Un SGBDA es un sistema de gestión de bases de datos (SGBD) que contiene la capacidad de definición, almacenamiento y gestión de reglas activas (Triggers). Además, de ejecutar lógica procedural (Procedimientos almacenados y Funciones).



Características

Módulos:

- **Programación de Reglas:** se encarga de la programación de las reglas de la base de datos. Estas reglas son programadas (planificadas) en la base de datos para mantener la consistencia y la integridad de los datos.
- **Gestor de Eventos:** encargado de la detección de los eventos (ej. Insert, Update, Delete).

Características

VENTAJAS DE UNA BD ACTIVA

- **Mayor independencia y consistencia.**
- **Centralización de la Información y comportamiento:** esto permite un mejor mantenimiento, ya que las reglas son almacenadas dentro de la Base de Datos, si es necesaria alguna modificación se hace sólo una vez en la BD en lugar de hacerlo en cada Programa.
- **Encapsulamiento de Procedimientos:** esto permite una mayor productividad ya que se pueden normalizar los procesos, sacando factor común de cierta lógica de los programas que se almacena una sola vez de forma centralizada. Permite la reutilización del código, ya que está disponible cada vez que se necesite.
- Minimiza también la transferencia de información.

Características

DESVENTAJAS DE UNA BD ACTIVA

- **Estandarización Escasa:** Si bien el Standard SQL 2003 define reglas generales para el encabezado el lenguaje interno de programación del trigger los SGBDA comerciales han realizado implementaciones muy diferentes.

- **Aumenta el consumo de recursos del nodo del SGBDA**: Al trasladar las operaciones al SGBDA, algunas operaciones que se resolvían desde las aplicaciones pasan a ser procesadas en el SGBDA aumentando el consumo de recursos de hardware dónde se aloje el mismo.

Resumen

- Introducción
- Características
- Paradigma ECA
- Reglas Activas
- Casos de estudio

Paradigma ECA

Las Reglas Activas siguen un modelo llamado “Paradigma ECA”, que es una sigla de **Evento** – Condición – Acción

Reacciona a los eventos que ocurren sobre los datos, evaluando condiciones que dependen de estos y ejecuta una acción cuándo la condición es verdadera.

Paradigma Evento – Condición – Acción

Eventos posibles:

- DML que cambian el estado de la BD (INSERT, UPDATE o DELETE)
- Consultas (SELECT)
- Cronograma (Ej. Viernes a las 5hs)
- Externos (Definidos por el usuario)

Paradigma Evento – Condición – Acción

Un **evento** consiste en una pareja del tipo: { <TOC> , <Tipo> }

Donde:

- **TOC** (Tiempo de ocurrecia): corresponde al punto en el tiempo cuando ocurre.
- **Tipo** de evento es la descripción o especificación del evento a detectar.

Ejemplos:

- { AFTER, INSERT }
- { BEFORE, UPDATE}

Paradigma Evento – Condición – Acción

Tiempo de ocurrencia

- **Inmediatas**, las acciones son ejecutadas:
  - ✓ **BEFORE** → Antes de…
  - ✓ **AFTER** → Después de…
  - ✓ **INSTEAD OF** → En lugar de…

  Respecto del evento que es monitoreado.

- **Diferidas**, la acción de la regla se ejecuta al final de una transacción, similar a AFTER, pero no necesariamente inmediatamente después.

Paradigma Evento – Condición – Acción

- Los tipos de eventos pueden ser situaciones dentro de la base de datos o sucesos en el ambiente. En el caso de las BD Relacionales hablamos de SELECT, INSERT, UPDATE y DELETE y para el caso de BD orientadas a objeto corresponderían las instrucciones de Creación, Llamado y Borrado de objetos.

- También puede ser considerado un tipo de evento un punto absoluto en el tiempo (“9 de julio de 2020 8:05 hs'') o puntos relativos o periódicos en el tiempo (“todos los domingos a las 6:30 hs.'').

Paradigma Evento – Condición – Acción

Eventos simple:

La mayoría de las veces las reglas se limitan a monitorear un solo evento.

Eventos compuestos:

Ocurren a partir de la ocurrencia de dos o más eventos utilizando un álgebra de eventos: conjunción, disyunción, negación, etc.
(Generalmente no está soportado por los SGBD comerciales)

Paradigma Evento – Condición – Acción

Historial de Eventos:

Se denomina historial de eventos al conjunto de todos los eventos sucedidos en el tiempo. La historia inicia desde el momento en que se define el primer tipo de evento en la base de datos.

Paradigma Evento – Condición – Acción

Condiciones:

La **condición** determina si la acción de la regla se debe ejecutar o no.

Una vez ocurre el evento disparador, se puede evaluar una condición (**es opcional**).

- **Si no se especifica** una condición, la acción se ejecutará **SIEMPRE** cuando suceda el evento.
- **Si se especifica** condición, la acción se ejecutará **SÓLO SI** la condición se **EVALÚA VERDADERO**.

Paradigma Evento – Condición – Acción

Acciones:

La **acción** a realizar puede ser una transacción sobre la base de datos o un programa externo que se ejecutará automáticamente.

Es un secuencia de sentencias SQL, que pueden estar inmersas en un lenguaje de programación integrado en el producto que se esté utilizando (por ejemplo, PL/SQL en Oracle)

Paradigma Evento – Condición – Acción

Las **acciones** pueden ser:

- Programas de manipulación de datos arbitrarios (SP / Funciones)
- Comandos transaccionales (DMLs) en si
- Comandos de activación de reglas (activación, desactivación)
- Llamadas a procedimientos externos (Funciones/Script Externas)
- Excepciones o errores

Paradigma Evento – Condición – Acción

Las **acciones** se dividen en:

- **Externas**: Se dan cuando son especificadas por aplicaciones, por ejemplo enviar un correo electrónico (*email*), imprimir una orden de compra, activar un dispositivo hardware.
- **Internas**: Son acciones de la base de datos, como un *insert, update, delete.*

Nota: En la practica vamos a ver que las acciones externas se pueden reemplazar por internas + Stored Procedure.

Paradigma Evento – Condición – Acción
Caso de estudio: punto de reposición

Si el stock de un producto baja de un cierto valor estipulado solicitar provisión.

Escenario: Toda aplicación que modifique el stock de algún producto hay que añadir código que compruebe si se baja del umbral para solicitar provisión.

La semántica está distribuida por las aplicaciones.

Posiblemente es una fuente de errores.

Paradigma Evento – Condición – Acción
Caso de estudio: punto de reposición

Si quisiéramos pasar este caso a un esquema activo.

Identificar cada elemento:

EVENTO	CONDICIÓN	ACCIÓN
???	???	???

Paradigma Evento – Condición – Acción
Caso de estudio: punto de reposición

Si quisiéramos pasar este caso a un esquema activo.

Identificar cada elemento:

EVENTO	CONDICIÓN	ACCIÓN
Modificación del stock	Stock < punto de pedido	Enviar un pedido al proveedor

Paradigma Evento – Condición – Acción
Caso de estudio: punto de reposición

- En pseudocódigo genérico:

```
ON evento IF condición THEN acción/es
```

Si se produce un evento y se cumple una determinada condición en la BD se realiza una acción.

- En pseudocódigo genérico:

```
ON update Producto(stock)
    IF Producto.stock < Producto.puntoPedido
    THEN INSERT INTO Pedidos …
```

Paradigma Evento – Condición – Acción
Caso de estudio: punto de reposición

- Código PLPGSQL:

```
CREATE TRIGGER trgReposicionPedidos
  AFTER UPDATE OF stock ON Productos
  FOR EACH ROW
  WHEN (new.stock < new.PuntoPedido)
  Execute Procedure fxEnviarPedidos();

CREATE FUNCTION fxEnviarPedidos() RETURNS TRIGGER AS $$
  DECLARE
  BEGIN
    Insert into Pedidos (productoCodigo, fecha, cantidadPedida)
         values (new.codigo, current_date, new.PuntoPedido * 3);
    RETURN Null;
 END;
$$Language PLpgSql;
```

Resumen

- Introducción
- Características
- Paradigma ECA
- Reglas Activas
- Casos de estudio

Reglas activas
Granularidad

Las reglas activas cuentan con dos niveles distintos de granularidad:

- **A nivel de instancia o fila**:

  La activación tiene lugar para cada tupla involucrada en la operación.

- **A nivel de instrucción o sentencia**:

  La activación tiene lugar sólo una vez para cada sentencia SQL. Las DMLs funcionan como eventos.

Reglas activas
Valores de Transición

Los **Valores de transición** se pueden considerar como variables locales que describen los cambios de estado realizados por una transacción:

- A nivel de instancia: Los valores de transición que afectan cada tupla son almacenados en dos objetos NEW y OLD.
- A nivel de instrucción: Los valores son almacenados en tablas INSERTED y DELETED

Nota: Estos conceptos son implementados en forma parcial o total por los SGBDA mediante objetos trigger

Reglas activas
Valores de Transición - INSTANCIA

Caso de Análisis a nivel de **Instancia** en **INSERT**

Empleado

dni	nombre	fecnac	sueldo
17626044	Torres, Maria	28/2/1975	80
24001719	Reyes, Miguel	14/6/1990	40
20320551	Rojas, Aldo	3/4/1982	60
19044333	Fuentealva, Elena	12/12/2002	50

"Empleado" tiene una Regla Activa/trigger de granularidad INSTANCIA.

Ejecutamos:

```
INSERT INTO persona VALUES (18325259,'RE ANDREA,'1992/3/2',58);
```

Filas Afectadas y valores de transición: (se dispara 1 instancia)

OLD	NEW
No existe	dni: 18325259, nombre: Re, Andrea, fecnac: 2/3/1992, sueldo: 58

Reglas activas
Valores de Transición - INSTANCIA

Caso de Análisis a nivel de **Instancia** en **UPDATE**

Empleado

dni	nombre	fecnac	sueldo
17626044	Torres, Maria	28/2/1975	80
24001719	Reyes, Miguel	14/6/1990	40
20320551	Rojas, Aldo	3/4/1982	60
19044333	Fuentealva, Elena	12/12/2002	50

"Empleado" tiene una Regla Activa/trigger de granularidad INSTANCIA.

Ejecutamos:

```
UPDATE persona SET sueldo = sueldo * 1.5 WHERE sueldo >= 60;
```

Filas Afectadas y valores de transición: (se disparan 2 instancias)

OLD

dni	nombre	fecnac	Old.sueldo
17626044	Torres, Maria	28/2/1975	80
20320551	Rojas, Aldo	3/4/1982	60

NEW

dni	nombre	fecnac	New.Sueldo
17626044	Torres, Maria	28/2/1975	120
20320551	Rojas, Aldo	3/4/1982	90

Reglas activas
Valores de Transición - INSTANCIA

Caso de Análisis a nivel de **Instancia** en **DELETE**

Empleado

dni	nombre	fecnac	sueldo
17626044	Torres, Maria	28/2/1975	80
24001719	Reyes, Miguel	14/6/1990	40
20320551	Rojas, Aldo	3/4/1982	60
19044333	Fuentealva, Elena	12/12/2002	50

"Empleado" tiene una Regla Activa/trigger de granularidad INSTANCIA.

Ejecutamos:

```
DELETE FROM persona WHERE sueldo <= 50;
```

Filas Afectadas y valores de transición: (se disparan 2 instancias)

OLD	NEW
dni | nombre | fecnac | sueldo 24001719 | Reyes, Miguel | 14/6/1990 | 40 19044333 | Fuentealva, Elena | 12/12/2002 | 50	No existe

Reglas activas
Valores de Transición - INSTRUCCIÓN

Caso de Análisis a nivel de **Instrucción** en **INSERT**

Empleado

dni	nombre	fecnac	sueldo
17626044	Torres, Maria	28/2/1975	80
24001719	Reyes, Miguel	14/6/1990	40
20320551	Rojas, Aldo	3/4/1982	60
19044333	Fuentealva, Elena	12/12/2002	50

"Empleado" tiene una Regla Activa/trigger de granularidad INSTRUCCIÓN.

Ejecutamos:

```
INSERT INTO persona VALUES (18325259,'RE ANDREA,'1992/3/2',58);
```

Filas Afectadas y valores de transición: (se dispara un solo Trigger)

DELETED	INSERTED
Vacío	dni: 18325259 · nombre: Re, Andrea · fecnac: 2/3/1992 · sueldo: 58

Reglas activas
Valores de Transición - INSTRUCCIÓN

Caso de Análisis a nivel de **Instrucción** en **UPDATE**

Empleado

dni	nombre	fecnac	sueldo
17626044	Torres, Maria	28/2/1975	80
24001719	Reyes, Miguel	14/6/1990	40
20320551	Rojas, Aldo	3/4/1982	60
19044333	Fuentealva, Elena	12/12/2002	50

"Empleado" tiene una Regla Activa/trigger de granularidad INSTRUCCIÓN.

Ejecutamos:

```
UPDATE persona SET sueldo = sueldo * 1.5 WHERE sueldo >= 60;
```

Filas Afectadas y valores de transición: (se dispara un solo Trigger)

DELETED

dni	nombre	fecnac	Old.sueldo
17626044	Torres, Maria	28/2/1975	80
20320551	Rojas, Aldo	3/4/1982	60

INSERTED

dni	nombre	fecnac	New.Sueldo
17626044	Torres, Maria	28/2/1975	120
20320551	Rojas, Aldo	3/4/1982	90

Reglas activas
Valores de Transición - INSTRUCCIÓN

Caso de Análisis a nivel de **Instrucción** en **DELETE**

Empleado

dni	nombre	fecnac	sueldo
17626044	Torres, Maria	28/2/1975	80
24001719	Reyes, Miguel	14/6/1990	40
20320551	Rojas, Aldo	3/4/1982	60
19044333	Fuentealva, Elena	12/12/2002	50

"Empleado" tiene una Regla Activa/trigger de granularidad INSTRUCCIÓN.

Ejecutamos:

```
DELETE FROM persona WHERE sueldo <= 50;
```

Filas Afectadas y valores de transición: (se dispara un solo Trigger)

DELETED	INSERTED
dni | nombre | fecnac | sueldo 24001719 | Reyes, Miguel | 14/6/1990 | 40 19044333 | Fuentealva, Elena | 12/12/2002 | 50	Vacío

Reglas activas
Selección de Reglas

Varias Reglas pueden estar en condiciones de ser disparadas al mismo tiempo, esto da lugar a un **conjunto conflicto**.

Las estrategias para seleccionar las reglas del conjunto conflicto se relacionan con prioridades:

- **Determinista**: Orden con prioridad numérica o alfabética.
- **No Determinista**: mediante prioridades explícitas, definidas directamente por el usuario cuando se crea la regla.

Reglas activas
Manipulación de reglas

La manipulación de Reglas se controla con sentencias especificas DDL:

- CREATE TRIGGER
- ALTER TRIGGER
- DELETE TRIGGER

NOTA: tenga en cuenta que la sentencia ALTER se puede utilizar para activar o desactivar la ejecución de una regla. Esto puede ser peligroso en un SGBDA en producción.

Reglas activas
Clasificación

Las Reglas Activas se pueden clasificar de la siguiente manera:

- **Propagación de Actualizaciones** para manejo de la redundancia, generalmente
- **Validadoras** de la consistencia
- Registración de **Pistas de auditoría**
- **Versionado de filas**
- **Mantenimiento de Vistas Materializadas**
- **Restauradoras** de la consistencia
- Manejo de **Valores por Defecto** de columnas y/o filas.

Reglas activas
Clasificación

REGLAS DE PROPAGACIÓN DE ACTUALIZACIONES

En un sistema de base de datos, su diseño se modela con la aplicación de relaciones **normalizadas**.

La aplicación de esta metodología en forma pura evita la redundancia, minimiza errores, sin embargo, usualmente algunos casos tienen baja performance.

Una alternativa que une ambos conceptos es modelar **datos redundantes**, por ejemplo de datos acumulados, pero con esta **redundancia controlada por reglas activas**.

El ejemplo mas claro de esto es el caso de análisis de transferencia bancaria con el atributo “saldo”.

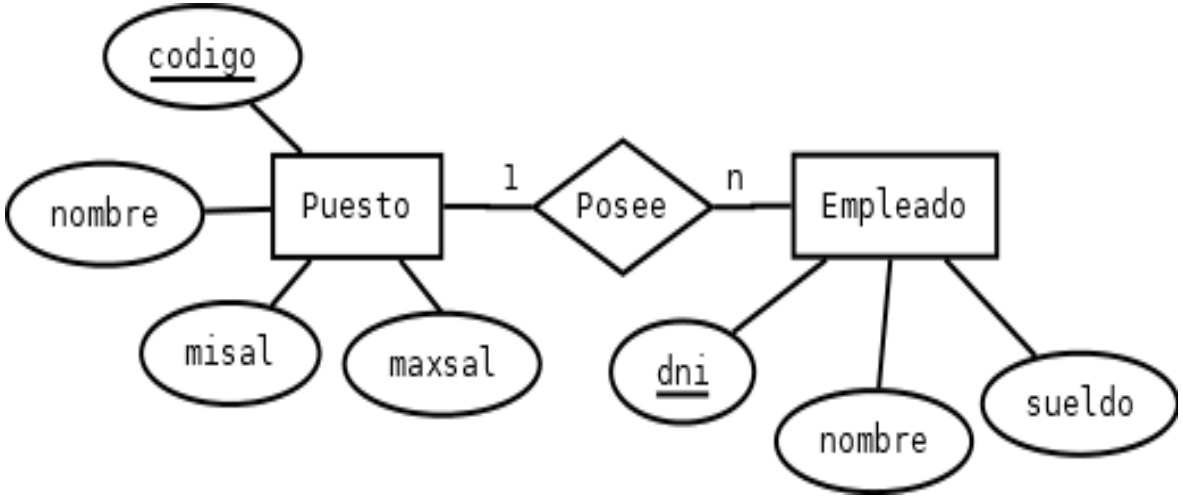
Estas reglas se han denominado reglas de administración de la redundancia controlada.

Reglas activas
Clasificación

REGLAS VALIDADORAS

Los sistemas de base de datos actuales permiten realizar validaciones simples en forma declarativa, por ejemplo la column-constraint "NOT NULL" o "UNIQUE", etc.

Pero determinadas operaciones complejas no se pueden realizar en forma declarativa, como por ejemplo, un control que plantea si un salario asignado a un puesto esta entre su valor máximo y mínimo, para ese puesto.



Reglas activas
Clasificación

REGLAS VALIDADORAS EJEMPLO

```
CREATE TRIGGER CompruebaSalario
BEFORE INSERT OR UPDATE OF sueldo, Puesto ON Empleado
FOR EACH ROW
DECLARE
    vminsal NUMBER;
    vmaxsal NUMBER;
    Salario_Fuera_Rango EXCEPTION;
BEGIN
    SELECT minsal, maxsal INTO vminsal, vmaxsal
      FROM Puesto
     WHERE codigo = :NEW.Puesto;

    IF (:NEW.sueldo < vminsal OR :NEW.Salario > vmaxsal) THEN
        RAISE Salario_Fuera_Rango;
    END IF;
END;
```

Reglas activas
Clasificación

REGISTRACIÓN DE PISTAS DE AUDITORÍA

Las transacciones activas en el SGBD tiene asociado el usuario que "disparo" la misma (el usuario de conexión), por lo que es posible controlar la ejecución de la misma de acuerdo al usuario, además la posibilidad de saber el usuario nos permite mantener información necesaria para auditorias.

Ej. Incorporar el Usuario y la Fecha Actual automáticamente al momento de realizar un UPDATE.

```
CREATE TRIGGER Actualizo_Clientes FOR Clientes
BEFORE UPDATE ON Clientes AS
BEGIN
   NEW.UltMod_Usuario = CURRENT_USER;
   NEW.UltMod_Fechahora = CURRENT_TIMESTAMP;
END;
```

Reglas activas
Clasificación

VERSIONADO DE FILAS

En todo sistema de aplicación existen un conjunto de tablas visibles que, mas allá de las restricciones DCL para su actualización, es conveniente registrar quien, a que hora, que función realizo y su estado anterior y posterior.

Vimos en el ejemplo anterior como realizar las primeras acciones, pero si tuviésemos una tabla idéntica a la original podríamos guardar completamente los valores anteriores a cada cambio en la tabla original en esta tabla que llamaríamos histórica.

Reglas activas
Clasificación

MANTENIMIENTO DE VISTAS MATERIALIZADAS

A diferencia de las vistas "normales" una vista materializada almacena físicamente los datos resultantes de ejecutar la consulta definida en la vista.

Este tipo de vistas materializadas realizan una carga inicial de los datos cuando se definen y posteriormente con una frecuencia establecida se actualizan los datos de la misma.

Con la utilización de vistas materializadas logramos aumentar el rendimiento de las consultas SQL además de ser un método de optimización a nivel físico en modelos de datos muy complejos y/o con muchos datos.

Una vez definida una vista materializada uno de los problemas que nos encontramos es el de la actualización de los datos. Para resolver este problema se usan Reglas Activas.

Reglas activas
Clasificación

REGLAS RESTAURADORAS DE LA CONSISTENCIA

En un modelo de base de datos las reglas de negocio se modelan como restricciones de este. Estas restricciones representan las propiedades del mundo real.

Cuando se intenta actualizar la base de datos y se presenta una restricción generalmente se rechaza la transacción que la ha provocado, devolviendo la base de datos al estado anterior a su ejecución.

Esta solución usualmente es poco satisfactoria. Una alternativa a este comportamiento consiste en que el sistema modifique el estado inconsistente, tomando una **decisión determinista**, de forma que se repare la violación provocada por la transacción de usuario respetando los cambios propuestos por ésta.

Reglas activas
Clasificación

REGLAS DE MANEJO DE VALORES POR DEFECTO

Existen un sin numero de situaciones que requieren que una atributo de una fila asuma un valor por defecto establecido por una regla de negocio, como casos concretos podemos plantear:

Administración de atributos autoincrementales, ej nro correlativo de socios, nro de remito de venta, etc.

Fecha del Sistema, Ejemplo fecha de registración, fecha de devolución de libro, etc.

Usuario que realizo la transacción (utilizando un usuario del sistema de aplicación como usuario de la base de datos), etc.

Estas reglas se han denominado reglas de administración de valores iniciales o valores por defecto de atributos o propiedades de las clases y tablas.

Resumen

- Introducción
- Características
- Paradigma ECA
- Reglas Activas
- Casos de estudio

Casos de estudio

Planteo: En un sistema de RRHH se desea automatizar una regla de negocio que compruebe el salario máximo presupuestado y el mínimo posible para cada puesto de una empresa.

Las tablas dispuestas por el responsable técnico del sistema son:

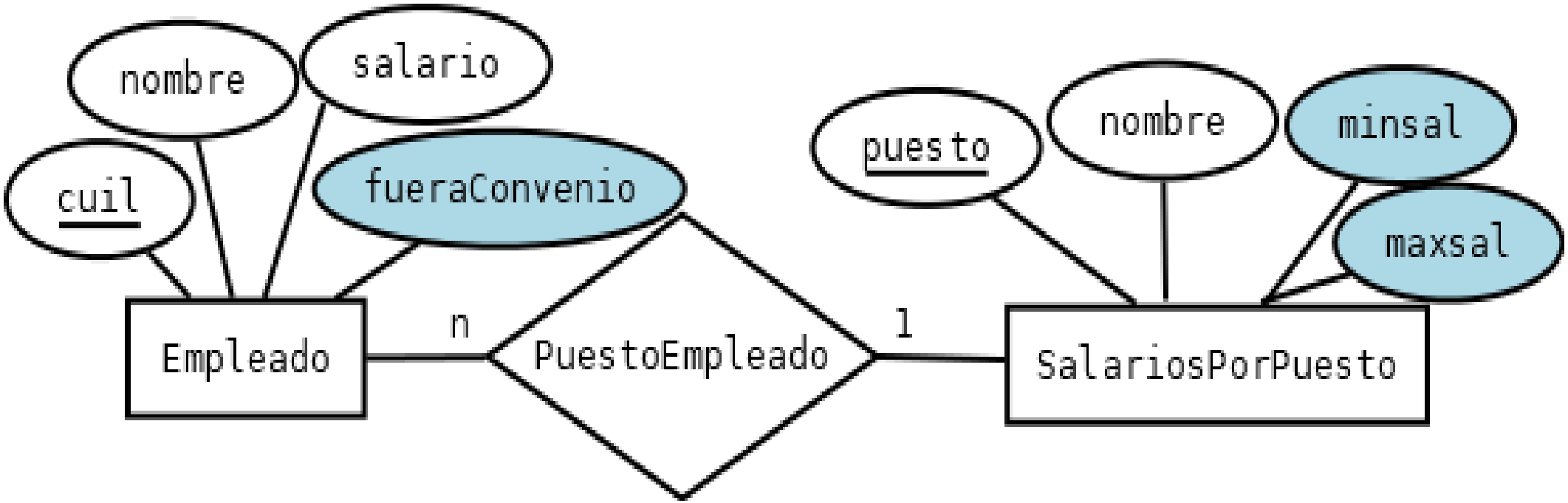
- Empleado(legajo, nombre, salario, puesto, fueraConvenio) Puesto FK con SalariosPorPuesto.puesto
- SalariosPorPuesto(puesto, nombre, minsal, maxsal)

Pedido: Se pide responda para la regla en cuestión:

a) Tabla
b) Tiempo de ocurrencia
c) Tipo de evento
d) Condición
e) Granularidad
f) Tipo de Trigger
g) Acciones en sql

Casos de estudio

Modelo



Casos de estudio

```
CREATE TRIGGER CompruebaSalario
BEFORE(b) INSERT(c) OR UPDATE(c) OF Salario, Puesto ON Empleado(a)
WHEN (NEW.fueraConvenio = false) (d)
FOR EACH ROW (e)
DECLARE
    minsal NUMBER;
    maxsal NUMBER;
    Salario_Fuera_Rango EXCEPTION;
BEGIN
    SELECT minsal, maxsal INTO minsal, maxsal
      FROM SalariosPorPuesto
     WHERE Puesto = :NEW.Puesto;

    IF (:NEW.Salario < minsal OR :NEW.Salario > maxsal) THEN
        RAISE Salario_Fuera_Rango;
    END IF;
END
```

f) Tipo de trigger: Validación compleja | g) Sentencias SQL: Entre BEGIN … END.