Tema 6 – Programación en Bases de Datos

Natalia Andrea Bueno Pizarro
Juan Guillermo Lalinde Pulido
ST0246 Bases de Datos

PROGRAMACIÓN EN BASES DE DATOS
Introducción

- Si bien las bases de datos tienen como función principal proporcionar persistencia y consulta, el hecho de centralizar la información facilita otras aplicaciones.
- En particular, si se puede almacenar código en la base de datos, éste estará disponible para cualquier aplicación que la utilice.
- Ventajas:
  - El código se puede reutilizar
  - Al actualizar el código, todas las aplicaciones se benefician
  - Garantiza que las reglas del negocio se respeten en todas las aplicaciones
  - Disminuye el tráfico de la red.

PROGRAMACIÓN EN BASES DE DATOS
Introducción

- Desventajas:
  - No hay un estándar para la programación
  - Consumen recursos del servidor
- Tipos de programación en bases de datos
  - Procedimientos almacenados (*stored procedures*)
  - Disparadores (*triggers*)
- Adicionalmente, se pueden desacoplar las aplicaciones de la base de datos utilizando una librería que medie en el proceso de solicitud de operaciones a la base de datos.

PROGRAMACIÓN EN BASES DE DATOS
Procedimientos almacenados

- Es una subrutina (función) que está disponible para todas las aplicaciones que acceden a un sistema de gestión de bases de datos particular.
- Siguiendo los principios de Codd, los procedimientos almacenados se almacenan en la base de datos como cualquier otro tipo de información.
- Sus usos más importantes son:
  - Validación
  - Control de acceso
  - Centralizar la lógica del negocio
- Los resultados normalmente son conjuntos, los cuales son recorridos utilizando *cursores*.
- El SQL, en sus versiones recientes, incluye expresiones como IF, WHILE, LOOP, REPEAT y CASE, lo que permite expresar algoritmos procedimentales.
- Normalmente se invocan mediante el comando CALL o EXECUTE.

PROGRAMACIÓN EN BASES DE DATOS
Procedimientos almacenados

- Su principal desventaja es que no hay lenguajes estándares para su implementación.

Database System	Implementation Language
Microsoft SQL Server	Transact-SQL y varios lenguajes del Framework .NET
Oracle	PL/SQL o Java
DB2	SQL/PL o Java
Informix	SPL
PostgreSQL	PL/pgSQL puede utilizar también sus propios lenguajes de funciones como pl/perl or pl/php
Firebird	PSQL (Fyracle también soporta un subconjunto del PL/SQL de Oracle)
MySQL	Su propio lenguaje, que se base en SQL:2003

PROGRAMACIÓN EN BASES DE DATOS
Disparadores

- Un disparador es un código que se ejecuta automáticamente en la base de datos como respuesta a un evento particular en una tabla o una vista.
- Normalmente los eventos a los que responden los disparadores son tres:
  - INSERT
  - UPDATE
  - DELETE
- Sus usos más importantes son:
  - Evitar cambios
  - Registrar cambios
  - Auditar cambios
  - Mejorar los cambios
  - Implementar reglas de negocio

PROGRAMACIÓN EN BASES DE DATOS
Disparadores

- Los eventos que disparan la acción pueden variar de manejador en manejador.
- En Oracle 9i se soportan disparadores que responden a cambios en el esquema de la base de datos.
- En SQL Server se tienen disparadores que responden a algunos eventos de cambio de esquema, como es crear una tabla nueva.
- En DB2 se soportan disparadores antes, después y en reemplazo del evento.
- Normalmente la integridad referencial se preserva en la base de datos utilizando disparadores que evitan que un registro sea eliminado si hay alguno otro registro que está relacionado con él mediante una clave foránea.

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- -----------------------------------------------------
-- Table `SIGUEME`.`Tbl_Identificacion_Personal`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGUEME`.`Tbl_Identificacion_Personal` (
  `Num_Id_Persona` INT NOT NULL AUTO_INCREMENT ,
  `Str_Nombres` VARCHAR(30) NULL ,
  `Str_Apellidos` VARCHAR(30) NULL ,
  `Str_FechaNacimiento` VARCHAR(12) NULL ,
  `Str_TarjetaIdentidad` VARCHAR(12) NULL ,
  `Str_Cedula` VARCHAR(12) NULL ,
  `Str_RegistroCivil` VARCHAR(12) NULL ,
  `Str_NumeroIDPersonal` VARCHAR(12) NULL ,
  `Dat_Fecha_Ingreso_Registro` DATETIME NULL ,
  `Num_id_Institucion` INT NULL ,
  `Str_Nombre_Completo` VARCHAR(60) NULL ,
  `Str_Id_Md5` VARCHAR(40) NULL ,
  `Str_Tipo_Sangre` VARCHAR(2) NULL ,
  `Num_id_genero` INT NULL ,
  `Num_id_Origen_Registro` INT NULL ,
  `Num_id_Estrato` INT NULL ,
  PRIMARY KEY (`Num_Id_Persona`))
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- -----------------------------------------------------
-- Table `SIGUEME`.`Tbl_Genero`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGUEME`.`Tbl_Genero` (
  `Num_id_Genero` INT NOT NULL ,
  `Str_Nombre_Genero` VARCHAR(2) NULL ,
  PRIMARY KEY (`Num_id_Genero`) )
```

PROGRAMACIÓN EN BASES DE DATOS

```
-- -----------------------------------------------------
-- Table `SIGUEME`.`Tbl_Identificacion_Personal`
-- -----------------------------------------------------


ALTER TABLE `SIGUEME`.`Tbl_Identificacion_Personal` ADD (
CONSTRAINT `fk_Tbl_identificacion_Personal_Num_id_Genero`
    FOREIGN KEY (`Num_id_genero`)
    REFERENCES `SIGUEME`.`Tbl_Genero` (`Num_id_Genero` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Tbl_Origen_Registro_Num_id_Origen_Registro`
    FOREIGN KEY (`Num_id_Origen_Registro` )
    REFERENCES `SIGUEME`.`Tbl_Origen_Registro` (`Num_id_Origen_Registro`
)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Tbl_Estrato_Num_id_Estrato`
    FOREIGN KEY (`Num_id_Estrato` )
    REFERENCES `SIGUEME`.`Tbl_Estrato` (`Num_id_Estrato` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

CREATE INDEX fk_Tbl_identificacion_Personal_Num_id_Genero ON
`SIGUEME`.`Tbl_Identificacion_Personal` (`Num_id_genero` ASC) ;

CREATE INDEX fk_Tbl_Origen_Registro_Num_id_Origen_Registro ON
`SIGUEME`.`Tbl_Identificacion_Personal` (`Num_id_Origen_Registro` ASC) ;

CREATE INDEX fk_Tbl_Estrato_Num_id_Estrato ON
`SIGUEME`.`Tbl_Identificacion_Personal` (`Num_id_Estrato` ASC) ;
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- ------------------------------------------------
-- CREATE USER
-- ------------------------------------------------


CREATE USER 'IE'@'localhost' IDENTIFIED BY 'IE123';

CREATE USER 'ES'@'localhost' IDENTIFIED BY 'ES123';

CREATE USER 'CF'@'localhost' IDENTIFIED BY 'CF123';

CREATE USER 'UR'@'localhost' IDENTIFIED BY 'UR123';
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- -----------------------------------------------------
-- Table `SIGUEME`.`Tbl_Genero`
-- -----------------------------------------------------
```

INSERTAR

```
DELIMITER $$

CREATE PROCEDURE `SIGUEME`.`SP_Nuevo_Genero` (P_Str_Nombre_Genero
varchar(30))

begin

      insert into Tbl_Genero (Str_Nombre_Genero) values
      (`P_Str_Nombre_Genero`);

end $$

DELIMITER;
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- -----------------------------------------------------
-- Table `SIGUEME`.`Tbl_Genero`
-- -----------------------------------------------------
```

MODIFICACIÓN.

```
DELIMITER $$

CREATE PROCEDURE `SIGUEME`.`SP_Mod_Genero` (IN   P_Num_id_Genero
int, IN P_Str_Nombre_Genero varchar (30))

begin

      Update Tbl_Genero set Str_Nombre_Genero
      =`P_Str_Nombre_Genero` where Num_id_Genero
      =`P_Num_id_Genero`;

end $$

DELIMITER;
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- -----------------------------------------------------
-- Table `SIGUEME`.`Tbl_Genero`
-- -----------------------------------------------------
```

ELIMINACIÓN

```
DELIMITER $$

CREATE PROCEDURE `SIGUEME`.`SP_Elim_Genero` (P_Str_Nombre_Genero varchar (30))

begin

      Delete from Tbl_Genero where  Str_Nombre_Genero =
      `P_Str_Nombre_Genero`;

end $$

DELIMITER;
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- -----------------------------------------------------
-- Table `SIGUEME`.`Tbl_Genero`
-- -----------------------------------------------------
```

LECTURA.

```
DELIMITER $$

CREATE PROCEDURE `SIGUEME`.`SP_Consulta_Genero` ()

begin

      Select Str_Nombre_Genero from Tbl_Genero;

end $$

DELIMITER;
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- -----------------------------------------------------
-- Table `SIGUEME`.`Tbl_Genero`
-- -----------------------------------------------------


LECTURAXID.

DELIMITER $$

CREATE PROCEDURE `SIGUEME`.`SP_Consulta_GeneroxID` (P_Num_id_Genero Int)

begin

Select Num_Id_Genero from Tbl_Genero where `P_Num_id_Genero` =
Num_id_Genero;

end $$

DELIMITER;
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- ----------------------------------------------------
-- PROCEDURE `SIGUEME`.`SP_Nuevo_Genero`
-- ----------------------------------------------------

Grant execute on procedure sigueme.SP_Nuevo_Genero
to admin@localhost
identified by 'admin';


-- ----------------------------------------------------
-- PROCEDURE `SIGUEME`.`SP_Mod_Genero`
-- ----------------------------------------------------

Grant execute on procedure sigueme.SP_Mod_Genero
to admin@localhost
identified by 'admin';
```

PROGRAMACIÓN EN BASES DE DATOS
Ejemplos

```
-- ----------------------------------------------------
-- PROCEDURE `SIGUEME`.`SP_Consulta_Genero`
-- ----------------------------------------------------

Grant execute on procedure sigueme.SP_Consulta_Genero
to admin@localhost
identified by 'admin';

grant execute on procedure sigueme.SP_Consulta_Genero
to IE@localhost
identified by 'insteducativa';

grant execute on procedure sigueme.SP_Consulta_Genero
to ES@localhost
identified by 'instsuperior';

grant execute on procedure sigueme.SP_Consulta_Genero
to CF@localhost
identified by 'instforma';

grant execute on procedure sigueme.SP_Consulta_Genero
to UR@localhost
identified by 'usuregistrado';
```