Data - Relational Model

10-02-2023

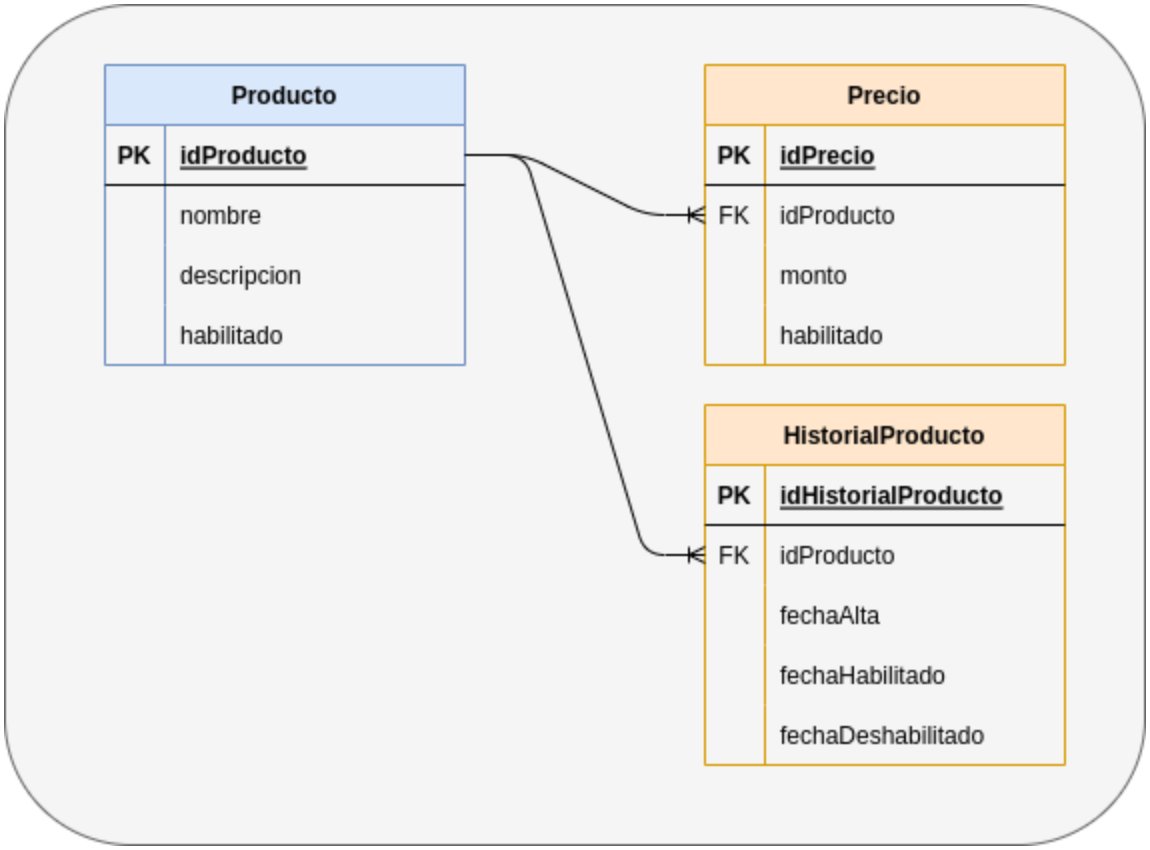
Leandro Baier

Metas

Temas a revisar:

1. Fundamentals of Database modeling
2. Knows the Concept of Primary Key y Foreign Key
3. Able to do Select/From/Where/OrderBy
4. Able to do Join Blocks
5. Able to do Create/Insert/Update/Delete-Truncate
6. Able to create SubQuery
7. Able to create Cursors
8. Able to create Function
9. Able to create Procedures
10. Able to create Package
11. Able to create Triggers

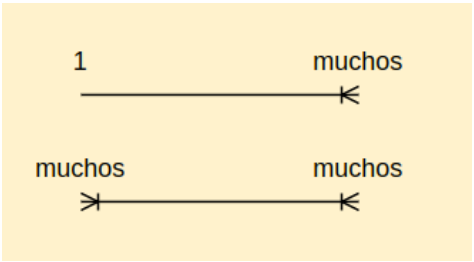
1. Fundamentals of Database modeling



2. Knows the Concept of Primary Key y Foreign Key

Primary Key: Es la llave principal de una tabla. Esta identifica unívocamente una tupla. Es única. Puede ser simple o compuesta por varias columnas.

Foreign Key: Es la forma de declarar las relaciones entre las tablas, para esto es necesario como mínimo 1 tabla (puede tener relación con sí misma), es cuando se utiliza la PK de una tupla para indicar la relación con otra tupla. Las constraint de las relaciones pueden ser 0, 1 y muchos teniendo en cuenta que esto es válido para cada lado de la relación, por ejemplo:



3. Able to do Select/From/Where/OrderBy

```
SELECT idProducto, nombre, descripcion
FROM Producto
WHERE habilitado = true;;
```

4. Able to do Join Blocks

```
SELECT prod.idProducto, pre.idPrecio, pre.monto
FROM Producto prod
INNER JOIN Precio pre
WHERE prod.habilitado = true
AND pre.habilitado = true;
```

5. Able to do Create/Insert/Update/Delete-Truncate

```
CREATE DATABASE matriz;
USE matriz;
CREATE TABLE Producto IF NOT EXIST (
        idProducto Integer PRIMARY KEY AUTOINCREMENT,
        nombre Varchar2(50) NOT NULL,
        descripcion Varchar2(200) NOT NULL,
        habilitado Boolean NOT NULL DEFAULT true
);
INSERT INTO Producto (idProducto, nombre, descripcion)
VALUES (1, 'Jabon', 'Jabon de mano');
UPDATE Producto
SET descripcion = 'Jabon de cuerpo'
WHERE idProducto = 1;
DELETE FROM Producto
WHERE idProducto != 1;
```

```
TRUNCATE TABLE Producto;
```

6. Able to create SubQuery

```
SELECT p.idProducto, p.nombre FROM Producto p
WHERE p.idProducto IN (SELECT idProducto FROM Producto pr
                       WHERE pr.habilitado = true);
```

7. Able to create Cursors

```
DELIMITER
BEGIN  $$
    DECLARE idPrecio Integer DEFAULT -1;
    DECLARE productoActual Producto;
    DECLARE productoActual
        CURSOR FOR
        SELECT * FROM Producto;


    OPEN productoActual;
    getIdPrecio: LOOP
            IF productoActual.idProducto >= 10 THEN
                idPrecio = SELECT idPrecio FROM precio p
                    WHERE idProducto = productoActual.idProducto
                    AND p.habilitado = true LIMIT 1;
                LEAVE getIdPrecio;
            END IF;
    END LOOP getIdPrecio;
    CLOSE productoActual;
END$$
DELIMITER
```

8. Able to create Function

```
DELIMITER
CREATE FUNCTION getIdPrecio(idProducto Integer) RETURNS Integer DETERMINISTIC
BEGIN
 DECLARE idPrecioProducto Integer;
  SELECT idPrecio INTO idPrecioProducto FROM precio p
                                        WHERE p.idProducto = idProducto
                                        AND p.habilitado = true LIMIT 1;
  RETURN idPrecio;
END
DELIMITER ;
```

9. Able to create Procedures

```
DELIMITER //
CREATE PROCEDURE getProductos()
BEGIN
        SELECT idProducto, nombre, descripcion  FROM producto WHERE habiltiado = true;
END //
DELIMITER ;
```

10. Able to create Package

```
CREATE OR REPLACE PACKAGE matriz_package AS pack
        FUNCTION getIdPrecio(idProducto Integer) RETURNS Integer;
        PROCEDURE getProductos();
END matriz_package;
```

11. Able to create Triggers

```
DELIMITER //
CREATE TRIGGER trigger_name  trigger_time trigger_event
        ON table_name FOR EACH ROW
BEGIN
        AFTER INSERT ON Producto FOR EACH ROW
        INSERT INTO HistorialProducto (idProducto, fechaAlta, fechaBaja)
        VALUES (NEW.idProducto, SYSDATE(),SYSDATE());
END //
DELIMITER;
```