111mil Programadores</>

Base de Datos

Apunte 1

Prof. Germán C. Basisty

basisty.german@gmail.com

LANÚS MUNICIPIO

FUNDACIÓN DIIEJ
Desarrollo para la Inclusión e Igualdad en el Empleo Juvenil

Índice de Contenidos

RDBMS: Concepto

Qué es una base de datos relacional

Una base de datos relacional es una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas, que permite acceder a los datos de maneras diferentes sin tener que reorganizar las tablas de la base.

Están constituidas por un conjunto de tablas (***relaciones***) que contienen datos provistos en categorías predefinidas. Cada tabla (***relación***) contiene una o más categorías de datos en columnas (***atributos***). Cada fila (***tupla***) contiene una instancia única de datos para las categorías definidas por las columnas (***atributos***).

Además de ser relativamente fáciles de crear y operar, una base de datos relacional tiene la importante ventaja de ser fácil de extender. Después de su creación, una nueva categoría de datos se puede añadir sin necesidad de que todos los elementos existentes sean modificados.

Qué es una RDBMS

Un sistema de gestión de bases de datos relacionales (***RDBMS***) es un conjunto de programas que permiten crear, actualizar y administrar una o varias bases de datos relacionales. Los **RDBMS** utilizan el *lenguaje estructurado de consultas* (***SQL*** por sus siglas en inglés, Structured Query Language) para manipular la base de datos.

Características de una base de datos relacional

Todas las bases de datos deben cumplir con las las *características* **ACID**:

- **A) Atomicidad:** Si cuando una operación consiste en una serie de pasos, o bien se ejecutan todos ellos o no se ejecuta ninguno, es decir, las transacciones son completas.

- **C) Consistencia:** (Integridad) Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de *Integridad* de la base de datos. La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido. "La Integridad de la Base de Datos nos permite asegurar que los datos son exactos y consistentes, es decir que estén siempre intactos, sean siempre los esperados y que de ninguna manera cambian ni se deformen. De esta manera podemos garantizar que la información que se presenta al usuario será siempre la misma."

- **I) Aislamiento:** Esta propiedad asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error. Esta propiedad define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes. El aislamiento puede alcanzarse en distintos niveles, siendo el parámetro esencial a la hora de seleccionar RDBMS.

- **D) Durabilidad:** (Persistencia). Esta propiedad asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema y que de esta forma los datos sobrevivan de alguna manera.

Cumpliendo estos 4 requisitos un sistema gestor de bases de datos puede ser considerado *ACID Compliant*.

Lenguaje SQL

SQL (por sus siglas en inglés Structured Query Language) es un lenguaje específico de las bases de datos relacionales, fundamentalmente basado en el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar información de bases de datos, así como hacer cambios en ellas.

SQL, que a veces se describe como un lenguaje declarativo, también incluye elementos procesales. Su alcance va desde la inserción de datos, consultas, actualizaciones y borrado, hasta la creación y modificación de esquemas y el control de acceso.

Las sentencias ***SQL*** se dividen en dos categorías:

- Lenguaje de definición de datos (*DDL*, data definition language)
- Lenguaje de manipulación de datos (*DML*, data manipulation language)

Lenguaje de definición de datos

Las sentencias **DDL** se utilizan para crear y modificar la estructura de las tablas así como otros objetos de la base de datos. Algunas de las instrucciones más comunes son:

- *CREATE* - para crear elementos en la base de datos.
- *ALTER* - modifica la estructura de elementos de la base de datos.
- *DROP* - borra elementos de la base de datos.

Lenguaje de manipulación de datos

Las sentencias **DML** son utilizadas para gestionar datos. Algunas de las instrucciones más comunes son:

- *SELECT* - para consulta de datos.
- *INSERT* - para inserción de datos.
- *UPDATE* - para actualización de datos preexistentes.
- *DELETE* - para eliminación de registros.
- *TRUNCATE* - para eliminación bruta de todos los registros.

DDL

Creación de relaciones

Para crear una relación en una base de datos relacional se debe llamar a la sentencia *CREATE* utilizando la siguiente sintaxis:

```
CREATE TABLE nombre_tabla (
        nombre_atributo         tipoDato,
        nombre_atributo         tipoDato,
        nombre_atributo         tipoDato
);
```

Donde **nombre_tabla** es el nombre de la relación, **nombre_atributo** es el nombre de cada atributo, y **tipoDato** es el tipo de dato del atributo.

Los tipos de datos soportados varían según el **RDBMS**, siendo los más comunes:

- integer (enteros)
- double (decimales)
- varchar(**n**) (textos de tamaño variable, donde **n** indica la longitud máxima del campo)
- date (fecha)
- time (hora)

A la hora de modelar una relación, pensar en ella como una clase pero (en principio) sin sus métodos, utilizando criterios de buenas prácticas de diseño similares.

Ejemplo

Crear una relación para dar persistencia a objetos de la clase Person. Los atributos de la misma son: dni, firstName, sureName y birthday:

```
CREATE TABLE person(
        dni             integer,
        first_name      varchar(20),
        surname         varchar(20),
        birthday        date
);
```

En el ejemplo anterior se instruye al **RDBMS** que cree una tabla llamada *person*, con los campos *dni* de tipo entero, *first_name* de tipo texto de tamaño variable de 20 caracteres de longitud, *surname* de tipo texto de tamaño variable de 20 caracteres de longitud, y *birthday* de tipo fecha.

Eliminación de relaciones

Para eliminar una relación se utiliza la instrucción *DROP*:

```
DROP TABLE nombre_tabla;
```

Donde *nombre_tabla* es el la relación que queremos eliminar

Ejemplo

Para eliminar la tabla *person* creada anteriormente:

```
DROP TABLE person;
```

Modificación de relaciones

Para modificar relaciones se utiliza la instrucción *ALTER*.

```
ALTER TABLE nombre_tabla MODIFICACIÓN A REALIZAR;
```

Donde *nombre_tabla* es el nombre de la relación a modificar. A continuación, se explican las operaciones básicas:

Agregar un atributo

```
ALTER TABLE nombre_tabla ADD COLUMN nombre_columna tipo_dato;
```

Ejemplo

```
ALTER TABLE person ADD COLUMN phone_number varchar(20);
```

Eliminar un atributo

```
ALTER TABLE nombre_tabla DROP COLUMN nombre_columna;
```

Ejemplo

```
ALTER TABLE person DROP COLUMN phone_number;
```

Renombrar un atributo

```
ALTER TABLE nombre_tabla RENAME nombre TO nuevo_nombre;
```

Ejemplo

```
ALTER TABLE person RENAME surname TO family_name;
```

Cambiar el tipo de dato de un atributo (cuando sea posible)

```
ALTER TABLE nombre_tabla ALTER COLUMN nombre_tabla
    SET DATA TYPE nuevo_tipodato;
```

Ejemplo

```
ALTER TABLE person ALTER COLUMN surname
    SET DATA TYPE text;
```

DML

Inserción de datos

Para insertar datos en una relación se utiliza la sentencia *INSERT* cuya sintaxis básica es la siguiente:

```
INSERT INTO nombre_tabla(nombre_atributo, nombre_atributo, nombre_atributo)
    VALUES(valor, valor, valor);
```

Donde **nombre_tabla** es el nombre de la relación en donde se van a insertar los datos, **nombre_atributo** es el nombre de cada atributo en el orden deseado, y **valor** es el dato a insertar en cada atributo según el orden especificado.
Es posible insertar más de una *tupla* a la vez con un único comando *INSERT*, separándolas con "," (coma).

Ejemplos

Continuando con la tabla creada con anterioridad:

- Inserción simple

```
INSERT INTO person(dni, first_name, surname, birthday)
    VALUES(30333444, 'Juan', 'Alvarez', '1990-09-05');
```

- Inserción múltiple

```
INSERT INTO person(dni, first_name, surname, birthday) VALUES
    (30333444, 'Juan', 'Alvarez', '1990-09-05'),
    (25111222, 'Andres', 'Martinez', '1977-12-23'),
    (35765432, 'Romina', 'Gonzalez', '1987-05-28');
```

Tener en cuenta que es posible omitir la lista de atributos (después del nombre de la relación) siempre y cuando se provean datos para *todos* los atributos y en el orden el el que fueron creados.

Consulta de datos

Para recuperar datos de una (en principio) relación utilizar la sentencia *SELECT* con la siguiente sintaxis:

```
SELECT
        atributo,
        atributo,
        atributo
FROM
        relación;
```

Esta consulta traerá los atributos listados de la relación especificada. Si deseamos recuperar la tupla completa podemos reemplazar el listado de atributos por el símbolo "*".

```
SELECT * FROM nombre_tabla;
```

Ejemplos

1) Recuperar todas las tuplas completas de la relación person

```
SELECT * FROM person;
```

2) Recuperar los campos first_name y surname de todas las tuplas de la relación person

```
SELECT
        first_name,
        surname
FROM
        person;
```

Lógica Proposicional

La cláusula *WHERE* se utiliza para agregarle lógica proposicional a una consulta y permite, en el caso de la sentencia *SELECT*, filtrar los resultados.

```
SELECT * FROM nombre_tabla WHERE atributo = valor;
```

Retornará las tuplas completas de la relación determinada siempre y cuando el o los atributos seleccionados cumplan con la condición de la cláusula *WHERE*.
Los operadores lógicos válidos son:

OPERADOR	CONDICIÓN
=	Igual a
<> o !=	Distinto a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
BETWEEN	Valor entre rangos
IN	Valor entre opciones
LIKE	Similar a. Se utiliza para evaluar strings (comodín: %)
ILIKE	Similar a, ignorando mayúsculas / minúsculas (comodín: %)

Operadores AND y OR

Los operadores *AND* y *OR* son usados para filtrar registros basados en más de una condición.

Operador AND

El operador **AND** muestra el registro si la primera condición y la segunda condición son verdaderas (todas).

Operador OR

El operador **OR** muestra el registro si la primera o la segunda condición es verdadera (alguna).

Ejemplos

1) Recuperar los campos dni, first_name y surname de las tuplas de la relación person donde la fecha de nacimiento sea anterior a 1990

```
SELECT
        dni,
        first_name,
        surname
FROM
        person
WHERE
        birthday < '1990-01-01';
```

2) Recuperar los campos dni y surname de las tuplas de la relación person donde el primer nombre sea "Juan", "Pedro" o "Ariel"

```
SELECT
        dni,
        surname
FROM
        person
WHERE
        first_name IN('Juan', 'Pedro', 'Ariel');
```

3) Recuperar las tuplas completas de la relación person donde el apellido sea "Perez" o el nombre sea "Adolfo"

```
SELECT
        *
FROM
        person
WHERE
        surname = 'Perez'
        OR first_name = 'Adolfo';
```

4) Recuperar las tuplas completas de la relación person donde el dni sea menor a 30000000 y la fecha de nacimiento esté entre los años 1960 y 1990

```
SELECT
        *
FROM
        person
WHERE
        dni < 30000000
        AND birthday BETWEEN '1960-01-01' AND '1990-01-01';
```

5) Recuperar las tuplas completas de la relación person donde el nombre sea similar a Ger…
sin tener en cuenta mayúsculas / minúsculas.

```
SELECT
        *
FROM
        person
WHERE
        name ilike 'ger%';
```

Resultado:
German | 25969467
gervasio | 13442567
GERARDO | 28777666

6) Recuperar las tuplas completas de la relación person donde el nombre sea similar a Ger…
teniendo en cuenta mayúsculas / minúsculas:

```
SELECT
        *
FROM
        person
WHERE
        name like 'ger%';
```

Resultado:
gervasio | 13442567

Ejercitación

1) Crear una solución para almacenar datos de automotores. Se deben tener en cuenta:

- Patente.
- Marca.
- Modelo.
- Cilindrada del motor.
- Tipo (Sedán o Coupe).

Cargar 20 registros. Escribir una consulta que me muestre las patentes de todos los vehículos cuya cilindrada sea superior a 1200cc.

2) Crear una solución para almacenar información de productos. Se debe tener en cuenta:

- Código.
- Rubro.
- Nombre del producto.
- Descripción.
- Unidad de medida.
- Precio unitario.

Cargar 20 registros. Escribir una consulta que me muestre los registros completos de todos los productos del rubro “almacen” cuyo precio sea inferior a $100.

3) Crear una solución para almacenar información de personas. Se debe tener en cuenta:

- Número de Pasaporte
- Nombre
- Apellido
- Nacionalidad
- Fecha de Nacimiento

Cargar 20 registros. Escribir una consulta que me muestre nombre, apellido y nacionalidad de todas las personas que no sean oriundas de Colombia y que hayan nacido entre los años 1995 y 2005.