Proyecto personal:

Base de datos Diccionario

Alejandro Miralles Ruiz
1 DAW

ÍNDICE

Enunciado

Se pide hacer una base de datos para guardar palabras. Se requiere que se cumplan los siguientes requisitos:

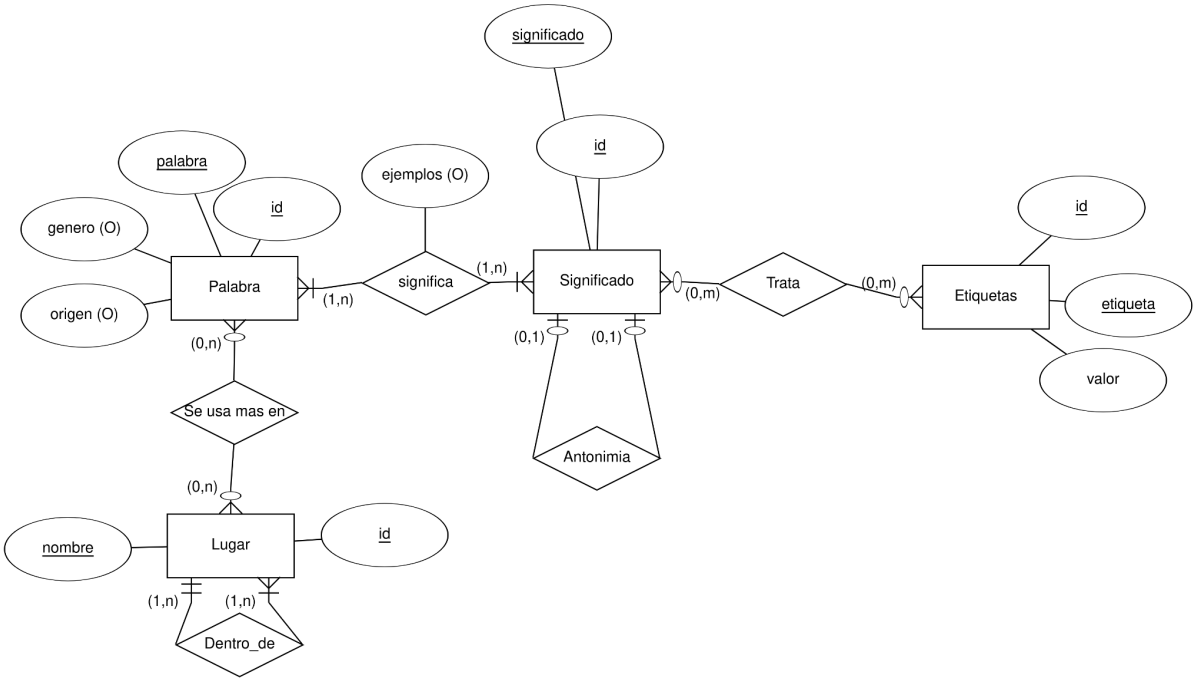
- Quiero guardar palabras con su respectivo género, etimología, lugar de uso más común, significado y con un ejemplo de su uso.

- Quiero poder hallar el significado de una palabra (con un ejemplo de su uso), sus sinónimos y sus antónimos.

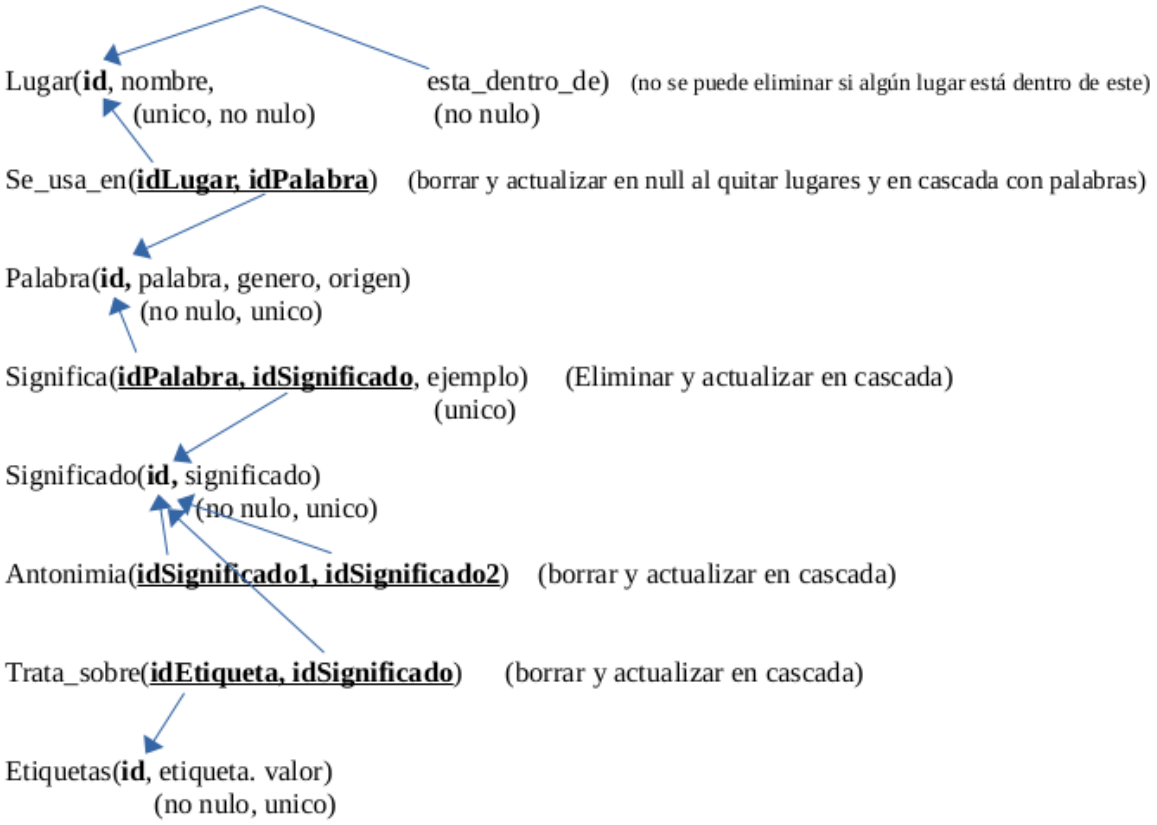
- Quiero poder hallar las palabras que se suelen usar en una determinada zona.

- Quiero poder encontrar palabras que se parezcan semánticamente a una palabra dada, sean sinónimas o no. Por ejemplo cráneo y cerebro.

Modelo Entidad-Relación



Modelo Relacional



Anotaciones: -Solo el lugar «mundo» no tendrá el campo «esta_dentro_de»
-Al modificar el «esta_dentro_de» de Lugar, hay que tener en cuenta que el nuevo «esta_dentro_de> NO está en un nivel más bajo que el anterior. Además, debe haber un trigger que pregunte si se está seguro de esa actualización (afecta al nivel de especificación de otros lugares)
Al borrar se mirará si hay nodos hijos e impedirlo de darse el caso

-En la tabla «Significa», ejemplo debe incluir la palabra de su correspondiente «idPalabra».

Base de datos: información

Junto a este documento readme se ha adjuntado la máquina virtual donde se encuentra la base de datos.
Para acceder a esta hay que entrar con el usuario “alumno”, cuya contraseña es “alumno”. Después sólo hay que abrir dbeaver, ya sea desde el enlace simbólico en el escritorio “dbeaver_Enlace” o accediendo a la carpeta “dbeaver” y luego abriendo el programa “dbeaver”.

La base de datos “diccionario” ya está vinculada con el dbeaver, pero por si acaso informo que su nombre, el nombre del propietario y su contraseña es “diccionario”.
A continuación muestro las tablas y funciones creadas, junto a los triggers.

Triggers

- Trigger_esta_dentro_de

```
create trigger trigger_esta_dentro_de before insert on lugar
    for each row
    execute procedure esta_dentro_de_valido();
```

-trigger_ids_inmodificables

```
create trigger trigger_ids_inmodificables before update or insert on Lugar,
Palabra, Significado, Etiquetas
   for each row
         execute procedure ids_inmodificables();
```

Los triggers sólo se pueden aplicar a una tabla, por lo que hay un trigger por cada tabla llamado "trigger_ids_inmodificables_*nombreTabla

-Significa_ejemplo_invalido

```
create trigger significa_ejemplo_invalido before insert or update on significa
   for each row
   execute procedure ejemplo_significa_valido();
```

- Lugar_update_estaDentroDe_valido

```
create trigger lugar_trigger_update_estaDentrode_valido before update on
lugar
   for each row
   execute procedure Modificacion_estaDentroDe_valido();
```

- Lugar_trigger_eliminacionDeLugar_valido

```
create trigger lugar_trigger_eliminacionDeLugar_valido before delete on lugar
   for each row
   execute procedure lugar_lugarEliminable();
```

Tablas

-Lugar

```
create table Lugar(
         id serial constraint lugar_clave primary key,
         nombre varchar(50) constraint lug_valido unique not null,
         esta_dentro_de bigint constraint dentro_de references Lugar on
      delete cascade on update cascade
)
```

-Se_Usa_En

```
create table se_Usa_En(
        idLugar bigint constraint referencia_Lugar_seUsaEn references Lugar
on delete set null on update set null,
        idPalabra bigint constraint referencia_Palabra_seUsaEn references
Palabra on delete cascade on update cascade,
        constraint claves_SeUsaEn primary key (idLugar, idPalabra)
)
```

-Palabra

```
create table Palabra(
        id serial constraint palabra_clave primary key,
        palabra varchar(20) constraint pal_unica unique not null,
        genero varchar(20) constraint genero_valido check
      (genero='Masculino' or genero='Femenino' or genero is null),
        origen varchar(20)
)
```

-Significa

```
create table Significa(
            idPalabra bigint constraint idpalabra_significa_referencia
          references palabra on delete cascade on update cascade,
            idSignificado bigint constraint
          idsignificado_significa_referencia references significado on
          delete cascade on update cascade,
            ejemplo text,
            constraint clavesPrincipales_significa primary key
          (idPalabra, idSignificado)
)
```

-Significado

```
create table Significado(
        id bigint constraint significado_clave primary key default
      nextval('next_id_significado'),
        significado text constraint significado_valido unique not null
)

create sequence next_id_significado start with 1;
```

- Antonimia

```
create table antonimia(
   idSignificado1 bigint constraint significado1_antonimia references
significado on delete cascade on update cascade,
   idSignificado2 bigint constraint significado2_antonimia references
significado on delete cascade on update cascade,
   constraint clavesPrincipales primary key (idSignificado1, idSignificado2)
   )
```

- Trata_sobre

```
create table trata_sobre(
        idSignificado bigint constraint idSignificado_trataSobre_references
      references significado on delete cascade on update cascade,
        idEtiqueta bigint constraint trataSobre_idEtiqueta_references
      references etiqueta on delete cascade on update cascade,
        constraint trataSobre_clavesPrincipales primary key (idSignificado,
      idEtiqueta)
)
```

- Etiqueta

```
create table Etiqueta(
        id serial constraint etiqueta_clave primary key,
        etiqueta varchar(50) constraint etiqu_unica unique not null,
        valor smallint constraint valor_etiqueta_valido check (valor between
      0 and 5)
)
```

Funciones para triggers

- Esta_dentro_de_valido()

```
create function esta_dentro_de_valido() returns trigger as $cos$
begin
   if ((not new.id=1) and new.esta_dentro_de is null)
        then raise exception 'Esta_dentro_de no puede ser nulo. Pon un
lugar válido';
   end if;
   return new;
end; $cos$
language 'plpgsql';
```

- Ids_inmodificables()

```
create or replace function ids_inmodificables() returns trigger as $cos$
declare
        cur refcursor;
        ultimoID bigint;
begin
        if TG_OP = 'UPDATE'
         then if (not new.id = old.id)
                  then raise exception 'Los IDS son invariables';
         end if;
         end if;
        if tg_table_name = 'lugar'
        then open cur for select tabla.id from lugar tabla order by 1
desc limit 1;
        elsif tg_table_name = 'palabra'
        then open cur for select tabla.id from palabra tabla order by 1
desc limit 1;
        elsif tg_table_name = 'significado'
        then open cur for select tabla.id from significado tabla order by
1 desc limit 1;
        elsif tg_table_name = 'etiqueta'
        then open cur for select tabla.id from etiqueta tabla order by 1
desc limit 1;
        end if;
        fetch cur into ultimoID;
        if TG_OP = 'INSERT'
        then if (not new.id > coalesce(ultimoID, 0))
                then close cur;
                raise exception 'El id nuevo debe ser el siguiente
número al anterior id';
        end if;
        end if;
        close cur;
        return new;
end; $cos$
language 'plpgsql';
```

- Ejemplo_significa_valido

```
create or replace function ejemplo_significa_valido() returns trigger as $cos$
declare
   cur cursor for select palabra from palabra where palabra.id =
new.idpalabra;
   palabraEjemplo varchar;
begin
        if (new.ejemplo is null)
        then return new;
        end if;
   open cur;
   fetch cur into palabraEjemplo;
   palabraEjemplo := lower(palabraEjemplo);
        if (lower(new.ejemplo) like '%' || palabraEjemplo || '%')
        then close cur;
         return new;
        end if;
         close cur;
        raise exception 'El ejemplo debe mostrar el uso de la palabra';
end; $cos$
language 'plpgsql';
```

- X_esta_dentro_de_Y

```
create or replace function X_esta_dentro_de_Y(hijo bigint, padre bigint)
returns bool as $cos$
declare
   cur cursor for select l.esta_dentro_de from lugar l where l.id = hijo;
   nuevoHijo bigint;
begin
        if (hijo=padre)
        then return true;
        end if;
         open cur;
         fetch cur into nuevoHijo;
        if (nuevoHijo is null)
         then close cur;
        return false;
        else
         close cur;
        return X_esta_dentro_de_Y(nuevoHijo, padre);
        end if;
end; $cos$
language 'plpgsql';
```

- Moficacion_estaDentroDe_valido

```
	create or replace function Modificacion_estaDentroDe_valido() returns
trigger as $cos$
declare
	curAntiguaDescendencia cursor for select l2.esta_dentro_de from
lugar l2 where l2.esta_dentro_de=old.id;
	supuestoHijoId bigint;
	curHijosActuales cursor for select * from lugar l where new.id =
l.esta_dentro_de;
begin
   open curHijosActuales;
   fetch curHijosActuales into supuestoHijoId;
	if ( supuestoHijoId  is null)
	then close curHijosActuales;
	return new;
	end if;
	 close curHijosActuales;
	open curAntiguaDescendencia;
	fetch curAntiguaDescendencia into supuestoHijoId;
	while supuestoHijoId is not null loop
	if (X_esta_dentro_de_Y(new.id, supuestoHijoId))
	then raise exception 'Se ha indicado que el lugar está dentro de un
lugar que está dentro del lugar.';
	end if;
	fetch curAntiguaDescendencia into supuestoHijoId;
	end loop;
	close curAntiguaDescendencia;
	return new;
end; $cos$
language 'plpgsql';
```

- Lugar_lugarEliminable

```
create or replace function lugar_lugarEliminable() returns trigger as $cos$
declare
   curHijosActuales cursor for select l.esta_dentro_de  from lugar l where
old.id=l.esta_dentro_de;
   idSupuestoHijo bigint;
begin
   open curHijosActuales;
   fetch curHijosActuales  into idSupuestoHijo;
         if (idSupuestoHijo is not null)
         then close curHijosActuales;
         raise exception 'No se puede eliminar dicho lugar porque otros
lugares están dentro de él';
         end if;
          close curHijosActuales;
         return old;
end; $cos$
language 'plpgsql';
```

Funciones para el usuario:

-buscar_significados(varchar)

```
create or replace function buscar_significados(varchar) returns
Table(palabra_seleccionado varchar, definicion text, ejemplo_uso text) as
$cos$
begin
         return query select $1, s2.significado, s.ejemplo from palabra p ,
significa s , significado s2
         where lower($1)=lower(p.palabra) and p.id = s.idpalabra and
s.idsignificado = s2.id ;
end; $cos$
language 'plpgsql';
```

-buscar_sinónimos(varchar)

```
create or replace function buscar_sinonimos(varchar) returns
Table(palabra_seleccionada varchar, sinonimo_encontrado varchar) as $cos$
begin
	return query select $1  as palabra, p.palabra as sinonimo
	from palabra p, significa s, (select s.idsignificado as definicion from
palabra p , significa s
				where lower($1)=lower(p.palabra) and p.id =
s.idpalabra)
				as significadoPalabras
	where significadoPalabras.definicion = s.idsignificado and s.idpalabra
= p.id and
		not lower(p.palabra) = lower($1)
	;
end; $cos$
language 'plpgsql';
```

- buscar_antonimos(varchar)

```
create or replace function buscar_antonimos(varchar) returns
Table(palabra_seleccionada varchar, antonimo varchar) as $cos$
begin
	return query select $1, p.palabra
	from palabra p, antonimia a, significa s, (select s.idsignificado
as definicion from palabra p , significa s
				where lower($1)=lower(p.palabra) and
p.id = s.idpalabra)
				as significadoPalabras
	where a.idsignificado1 = significadoPalabras.definicion and
a.idsignificado2 = s.idsignificado
				and s.idpalabra = p.id
	;
end; $cos$
language 'plpgsql';
```

- Buscar_palabras_similares(varchar)

```
create or replace function buscar_palabras_similares(varchar) returns
Table(palabra_seleccionada varchar, palabra_similar varchar) as $cos$
begin
	return query select $1, p.palabra from palabra p,  (
	select s.idpalabra as palabrasimilar from significa s, trata_sobre t,
etiqueta e, (
		select t2.idetiqueta as etiquet from palabra p2, significa s2,
trata_sobre t2
			where lower(p2.palabra) = lower($1) and p2.id =
s2.idpalabra and s2.idsignificado = t2.idsignificado
		) as etiquetasDeLaPalabra
		where s.idsignificado  = t.idsignificado and t.idetiqueta =
etiquetasDeLaPalabra.etiquet
		and e.id = t.idetiqueta
		group by s.idpalabra having sum(coalesce(e.valor, 0)) >= 5
	) as palabrassimilares
	where p.id = palabrassimilares.palabrasimilar and not lower(p.palabra)
= lower($1);
end; $cos$
language plpgsql;
```

-Buscar_palabras_por_lugar(varchar)

```
create or replace function buscar_palabras_por_lugar(varchar) returns
Table(lugar_seleccionado varchar, palabra_usada varchar) as $cos$
begin
	return query select $1, p.palabra from palabra p, se_usa_en s,
			(select l.id from lugar l,
				(select l2.id from lugar l2
					where l2.nombre = $1) as lugarPalabra
			where x_esta_dentro_de_y(lugarPalabra.id, l.id)
			) as lugaresValidos
	where lugaresValidos.id= s.idlugar and s.idpalabra = p.id;
end; $cos$
language 'plpgsql';
```

-id_Lugar_Correspondiente(varchar)

```
create or replace function id_Lugar_Correspondiente(localizacion varchar)
returns bigint as $cos$
declare
   resultado bigint;
   cur cursor for select l.id from lugar l where l.nombre= $1;
begin
   fetch cur into resultado;
   return cur;
end; $cos$
language 'plpgsql';
```