Procedimientos almacenados

Segunda entrega 04/09/2019

Ruta en GitLab: /Actividades/BD01DPA/Procedimientos almacenados

La base de datos del SGLA contiene algunos procedimientos almacenados en lenguaje Informix SPL. Algunos de ellos son utilizados por el programa, mientras que otros existen únicamente para facilitar el uso y manipulación manual de la base de datos. A continuación se listan y detallan.

Crear_lugar(nombrel varchar(100), pos_x float, pos_y float, tipo varchar(15), capacidad integer, creador integer) -> integer

crear_lugar es un procedimiento almacenado que crea un lugar en el sistema a partir de la información dada y devuelve su ID. Realiza ciertas verificaciones con la información, actualmente verifica que la capacidad sea un número positivo y que el tipo de lugar que está siendo creado no sea una zona o subzona, los cuales deben ser creados con el procedimiento crear_subzona.

- Nombrel: Nombre del lugar a ser creado.
- Pos_x, pos_y: coordenadas del lugar a ser creado
- Tipo: Tipo de lugar a ser creado (“Puerto”, “Patio”, “Establecimiento”)
- Capacidad: capacidad del lugar (> 0)
- Creador: usuario creador del lugar
- Devuelve: el ID del lugar creado (<0 si no fue creada)

```
create function crear_lugar(nombrel like lugar.nombre, pos_x like lugar.geox, pos_y
like lugar.geoy, tipo like lugar.tipo, capacidad like lugar.capacidad, creador like
lugar.usuariocreador)
returning integer
       DEFINE lugarid int;
       IF capacidad < 1 THEN
           return -1;
       END IF
       IF tipo not in ('Puerto', 'Patio', 'Establecimiento') THEN
           return -2;
       END IF
       insert into lugar
       (idlugar, nombre, geox, geoy, capacidad, usuariocreador, tipo, fecharegistro)
       values
       (0,       nombrel,pos_x,pos_y,capacidad, creador,        tipo, current);
       select dbinfo('sqlca.sqlerrd1') into lugarid from systables where tabid=1;
       return lugarid;
end function;
```

crear_subzona(nombrez varchar(100), enLugar integer, capacidad integer) -> integer

crear_subzona es un procedimiento almacenado que crea una subzona en un lugar del sistema a partir de la información dada y devuelve su ID. Verifica que el lugar donde está siendo creada la subzona sea de tipo ‘zona’, exista, y su capacidad no supere al total del lugar.

- Nombrez: nombre de la subzona
- enLugar: ID de la zona a la que pertenece esta subzona
- Capacidad: capacidad de la subzona
- Devuelve: ID de la subzona creada (<0 si no fue creada)

```
create function crear_subzona(nombrez like lugar.Nombre, enLugar like lugar.IDLugar,
capacidadz int)
   returning integer
    DEFINE existelugar, capacidadlugar, creador, lugarid int;
    DEFINE gx, gy float;
    SELECT count(*), capacidad, geox, geoy, usuariocreador
    into existelugar, capacidadlugar, gx, gy, creador
    from lugar where idlugar=enLugar and tipo='Zona'
    group by idlugar, capacidad, geox, geoy, usuariocreador;
    IF existelugar < 1 THEN
        return -1;
    END IF;
    IF capacidadlugar < capacidadz THEN
        return -1;
    END IF;
    IF gx is null or gy is null THEN
    END IF;
    insert into lugar(idlugar, nombre, capacidad, geox, geoy,
            usuariocreador, tipo,fechaRegistro)
        values(0, nombrez, capacidadz, gx, gy, creador, 'Subzona',current);
    select dbinfo('sqlca.sqlerrd1') into lugarid from systables where tabid=1;
    insert into incluye values(lugarid, enLugar);
    return lugarid;
end function;
```

crear_zona(nombrez varchar(100), enLugar integer, capacidad integer) -> integer

crear_zona es un procedimiento almacenado que crea una zona en un lugar del sistema a partir de la información dada y devuelve su ID. Verifica que el lugar donde está siendo creada la zona sea de tipo 'Puerto' o 'Patio', exista, y su capacidad no supere la capacidad del lugar.

- Nombrez: nombre de la zona a ser creada
- enLugar: ID del lugar al que pertenece esta zona
- capacidad: capacidad de la zona
- Devuelve: ID de la zona creada (<0 si no fue creada)

```
create function crear_zona(nombrez like lugar.Nombre, enLugar like lugar.IDLugar,
capacidadz int)
   returning integer
    DEFINE existelugar, capacidadlugar, creador, lugarid int;
    DEFINE gx, gy float;
    SELECT count(*), capacidad, geox, geoy, usuariocreador
    into existelugar, capacidadlugar, gx, gy, creador
    from lugar where idlugar=enLugar and tipo in ('Puerto', 'Patio')
    group by idlugar, capacidad, geox, geoy, usuariocreador;
    IF existelugar < 1 THEN
        return -1;
    END IF;
    IF capacidadlugar < capacidadz THEN
        return -1;
    END IF;
    insert into lugar(idlugar, nombre, capacidad, geox, geoy,
            usuariocreador, tipo,fechaRegistro) values(0, nombrez, capacidadz, gx,
gy, creador, 'Zona', current);
    select dbinfo('sqlca.sqlerrd1') into lugarid from systables where tabid=1;
    insert into incluye values(lugarid, enLugar);
    return lugarid;
end function;
```

zonas_en_lugar(lugarid integer) -> {(integer, varchar(100), integer)}

zonas_en_lugar es un procedimiento almacenado que devuelve un conjunto de tuplas con la información principal sobre las zonas en un lugar (ID, nombre, capacidad). Leer el anexo sobre funciones “generadoras”.

- Lugarid: lugar cuyas zonas quiere recuperar
- Devuelve: (id, nombre, capacidad) [múltiples]

```
create function zonas_en_lugar(lugarid like lugar.idlugar)
   returning integer, varchar(100), integer
    DEFINE idz integer;
    DEFINE nmz varchar(100);
    DEFINE cpz integer;
    FOREACH cursor1 FOR
      select lugar.idlugar, nombre, capacidad
      into idz, nmz, cpz
      from lugar inner join
      (select menor as idlugar from incluye
       start with mayor=lugarid
       connect by prior menor=mayor) as children on lugar.idlugar=children.idlugar
      where lugar.tipo="Zona"
      return idz, nmz, cpz WITH RESUME;
    END FOREACH;
end function;
```

subzonas_en_zona(lugarid integer, zonaid integer) -> (integer, varchar(100), integer)

subzonas_en_zona es un procedimiento almacenado que devuelve un conjunto de tuplas con la información principal de las subzonas en una zona de un lugar dado. Leer el anexo sobre funciones "generadoras".

- Lugarid: ID del lugar al que pertenece la zona cuyas subzonas quieren ser devueltas
- Zonaid: ID de la zona cuyas subzonas quieren ser devueltas
- Devuelve: (id, nombre, capacidad) [múltiples]

```
create function subzonas_en_zona(lugarid like lugar.idlugar, zonaid like
lugar.idlugar)
   returning integer, varchar(100), integer
    DEFINE idz integer;
    DEFINE nmz varchar(100);
    DEFINE cpz integer;
    FOREACH cursor1 FOR
      select lugar.idlugar, nombre, capacidad
      into idz, nmz, cpz
      from lugar inner join
      (select menor as idlugar from incluye
       start with mayor=lugarid and menor=zonaid
       connect by prior menor=mayor) as children on lugar.idlugar=children.idlugar
      where lugar.tipo="Subzona"
      return idz, nmz, cpz WITH RESUME;
    END FOREACH;
end function;
```

subzonas_en_lugar(lugarid integer) -> (integer, varchar(100), integer)

subzonas_en_lugar es un procedimiento almacenado que devuelve un conjunto de tuplas con la información principal de las subzonas en un lugar dado. Leer el anexo sobre funciones “generadoras”.

- Lugarid: ID del lugar cuyas subzonas quieren ser devueltas.
- Devuelve: (ID, nombre, capacidad) [múltiples]

```
create function subzonas_en_lugar(lugarid like lugar.idlugar)
   returning integer, varchar(100), integer
    DEFINE idz integer;
    DEFINE nmz varchar(100);
    DEFINE cpz integer;
    FOREACH cursor1 FOR
      select lugar.idlugar, nombre, capacidad
      into idz, nmz, cpz
      from lugar inner join
      (select menor as idlugar from incluye
       start with mayor=lugarid
       connect by prior menor=mayor) as children on lugar.idlugar=children.idlugar
      where lugar.tipo="Subzona"
      return idz, nmz, cpz WITH RESUME;
    END FOREACH;
end function;
```

ocupacion_en_lugar(lugarid as integer) -> integer

ocupacion_en_lugar es un procedimiento almacenado que devuelve la ocupación actual de un lugar dado. En caso de ser una subzona, el procedimiento es trivial. En caso de no serlo, llama a subzonas_en_lugar(lugarid) y luego retorna la suma de cantidad de posiciones ocupadas por cada una.

- Lugarid: lugar cuya ocupación quiere ser devuelta
- Devuelve: ocupación del lugar

```
create function ocupacion_en_lugar(lugarid like lugar.idlugar)
   returning integer
    DEFINE tipo varchar(15);
    DEFINE ocup integer;
    select lugar.tipo into tipo from lugar where idlugar=lugarid;
    IF tipo <> "Subzona" THEN
       select count(*)  into ocup
       from posicionado where hasta is null
       and idlugar in (select unnamed_col_1 from table(subzonas_en_lugar(lugarid)));
    ELSE
       select count(*) into ocup
       from posicionado where hasta is null and idlugar = lugarid;
    END IF;
    return ocup;
end function;
```

MaximoAncestro(lugarid as integer) -> integer

Maximo ancestro es un procedimiento amcenado encargado de devolvernos el id del lugar del id subzona o id de la subzona que se le pasa por parámetro. Siendo extramandamente útil para evitar el uso del connect by o derivados.

```
CREATE FUNCTION maximo_ancestro(lugarid LIKE lugar.idlugar)
    returning integer
    define ttx int;
    IF lugarid NOT IN (SELECT menor FROM incluye) THEN
        RETURN lugarid;
    END if;
    select min(mayor) into ttx from incluye start with menor=lugarid connect by menor
 = prior mayor;
    return ttx;
END FUNCTION;
```

subzonas_en_lugar_por_nombre(lugarnombre varchar(100)) -> (integer, varchar(100), integer)

subzonas_en_lugar_por_nombre es un procedimiento almacenado que devuelve un conjunto de tuplas con información sobre las subzonas en el lugar dado. Leer el anexo sobre funciones generadoras. Su funcionamiento es equivalente al de subzonas_en_lugar pero filtrando según nombre en lugar de según ID.

- Lugarnombre: nombre del lugar cuyas subzonas quieren ser devueltas
- Devuelve: (ID, nombre, capacidad) [múltiples]

```
create function subzonas_en_lugar_por_nombre(lugarnombre like lugar.nombre)
   returning integer, varchar(100), integer
    DEFINE idz integer;
    DEFINE nmz varchar(100);
    DEFINE cpz integer;
    DEFINE lugarid integer;
    select lugar.idlugar into lugarid from lugar where nombre=lugarnombre;
    FOREACH cursor1 FOR
      select lugar.idlugar, nombre, capacidad
      into idz, nmz, cpz
      from lugar inner join
      (select menor as idlugar from incluye
       start with mayor=lugarid
       connect by prior menor=mayor) as children on lugar.idlugar=children.idlugar
      where lugar.tipo="Subzona"
      return idz, nmz, cpz WITH RESUME;
    END FOREACH;
end function;
```

cerrar_lote(loteid integer) -> boolean

cerrar_lote es un procedimiento almacenado que cierra un lote con la ID dada sí y solo sí todos los vehículos que lo integran han tenido informes en el lugar de origen.

- Loteid: ID del lote que se quiere cerrar
- Devuelve: ‘t’ en caso de que el lote haya sido cerrado, ‘f’ en caso contrario

```
create function cerrar_lote(loteid like lote.idlote)
    returning boolean
        DEFINE unverif boolean;
    select count(*) > 0
    into unverif
    from integra
    inner join vehiculo on integra.idvehiculo=vehiculo.idvehiculo and
integra.lote=loteid and integra.invalidado='f'
    inner join lote on integra.lote=lote.idlote
    left join informedanios on informedanios.idvehiculo=vehiculo.idvehiculo and
informedanios.idlugar=lote.origen
    where informedanios.id is null;
    IF unverif THEN
    return 'f';
    ELSE
    update lote set estado="Cerrado" where idlote=loteid;
    return 't';
    END IF;
end function;
```

ANEXO: Funciones generadoras

Existen en la base de datos un conjunto de funciones “generadoras”, o sea que devuelven más de un valor utilizando devoluciones con resume, esto es, la función devuelve una tupla pero no termina su ejecución, sino que queda en modo de espera a que el gestor le pida otra tupla, como los generadores de Python o las funciones async/await de Rust. La manera de pedir todas las tuplas de resultado es llamar, en SQL o en otra función, *table(funcion(args…))*, lo cual devolverá una tabla sin nombre cuyas columnas se llaman *unnamed_col_1, unnamed_col_2, …, unnamed_col_n*. Hacen uso de la función de conexión recursiva de Informix (connect by), por lo cual son fácilmente escalables si a futuro es necesario representar lugares con organización más profunda (sub-sub-zonas, etc).