Firebird White Paper

Firebird 3 Stored Functions

Fikret Hasovic, May 2021

Firebird 3 introduced very neat feature: the ability to write scalar functions in procedural SQL. Write them, commit them and they'll be declared to your database just as though they were external functions (UDFs). They have been named "stored functions", to distinguish them from "internal functions" (the functions that are built into Firebird's SQL language) and "external functions", functions that are loaded from an external library on demand, after being declared inside the database.

The syntax is very similar to stored procedures, although not exactly the same. There's a header, where the inputs and the output are declared, along with any local variables. The action happens inside a BEGIN...END block and the keyword RETURN sends the result back.

Suppose you need a function that takes two GPS coordinates (longitude and latitude) as an argument and returns a distance (sphere surface distance), e.g.,

```
create or alter function F_DIST (
    LON1 double precision,
    LAT1 double precision,
    LON2 double precision,
    LAT2 double precision)
returns double precision
as
declare variable LONX1 double precision;
declare variable LATX1 double precision;
declare variable LONX2 double precision;
declare variable LATX2 double precision;
declare variable DISTANCE double precision;
begin
  LONX1=pi()*LON1/180;
  LATX1=pi()*LAT1/180;
  LONX2=pi()*LON2/180;
  LATX2=pi()*LAT2/180;

  DISTANCE = ACOS(SIN(LONX1)*SIN(LONX2)+COS(LONX1)*COS(LONX2)*COS(LATX2-LATX1))*6378.137;
  return DISTANCE;
end
```

This is very similar to a stored procedure we created earlier to do the same job:

```
create or alter procedure DIST (
    PLZ1 varchar(5),
    PLZ2 varchar(5))
returns (
    DISTANCE double precision)
as
declare variable LON1 double precision;
declare variable LAT1 double precision;
declare variable LON2 double precision;
declare variable LAT2 double precision;
declare variable LONX1 double precision;
declare variable LATX1 double precision;
declare variable LONX2 double precision;
declare variable LATX2 double precision;
begin
  select address.lon, address.lat
  from address where address.plz=:plz1
  into :LON1, LAT1;
  select address.lon, address.lat
  from address where address.plz=:plz2
  into :LON2, LAT2;

  LONX1=pi()*LON1/180;
  LATX1=pi()*LAT1/180;
  LONX2=pi()*LON2/180;
  LATX2=pi()*LAT2/180;

  distance = ACOS(SIN(LONX1)*SIN(LONX2)+COS(LONX1)*COS(LONX2)*COS(LATX2-LATX1))*6378.137;
  suspend;
end
```

But, to duplicate the above stored procedure's functionality, we can write another stored procedure, such as:

```
create or alter procedure MY_DIST (
    PLZ1 varchar(5),
    PLZ2 varchar(5))
returns (
    DISTANCE double precision)
as
begin
  select F_DIST(p1.lon, p1.lat, p2.lon, p2.lat)
  from address p1, address p2
  where (p1.plz=:plz1) and (p2.plz=:plz2)
  into :distance;
  suspend;
end
```

That was easy! But Firebird 3 will let us run subroutines inside both stored functions and stored procedures, if needed. As you can see, user-defined stored functions can be used anywhere you would use a built-in or external scalar function. We are already referring to external functions as "legacy UDFs".

IBEBlock

If you have never used IBEBlock, it's time to take a look. IBEBlock is a set of DDL, DML and other statements that are executed on the server and on the client side, and which include some specific constructions applicable only in IBExpert or IBEScript (excluding the free versions of these products), independent of the database server version.

IBEBlock is an extremely complex and capable piece of software, which can help you to increase your productivity and save both time and money spent on your projects.

Here is a very nice example of how to use the Google Maps API to check addresses, so you can eliminate a lot of typos or completely wrong addresses, and it can be extremely cool to have the exact GPS location for finding, for example, companies close to or in the region of a specific company (typically a problem for salesmen, looking for the target of the next visit nearby).

```
execute ibeblock
as
begin
  try
    DB = ibec_CreateConnection(__ctFirebird,
                               'DBName="MYADDRESSBOOK.FDB";
                               ClientLib=fbclient.dll;
                               User=SYSDBA; Password=masterkey; SqlDialect=3;');
    i=0;

    use db;

    select count(*) from address where adr is null into anz;
    select extract(hour from current_time) from rdb$database into hr;
    if ((anz=0) or (hr between 4 and 9)) then  sleep(3600000);
    else
    begin
      ibec_progress(anz);
      for select
          address.id,
          address.street,
          address.city,
          address.state,
          address.zip
          from address
          order by id
          into id, street, city, state, zip
      do
      begin
        i=i+1;
        street=replace(street,' ','%20');
        city=replace(city,' ','%20');
        cmd=ibec_AnsiStringToUTF8('wget1 -O "D:\temp\'||id||'.json"
"https://maps.googleapis.com/maps/api/geocode/json?address='||street||'%20'||city||'&key=YOUR_API
_KEY" --no-check-certificate');
        ibec_exec(cmd,'','');
```

```
        s = ibec_LoadFromFile('d:\temp\'||id||'.json');
        if (ibec_pos('exceeded',s)>0) then
        begin
          commit;
          ibec_Progress('exceeded limit, wait an hour');
          ibec_sleep(3600000);
        end
        else
        begin
          p=ibec_pos('formatted_address',s)+22;
          if (p>0) then
          begin
            s=ibec_Copy(s,p,500);
            p2=ibec_pos('"',s)-1;
            sadr=ibec_Copy(s,1,p2);
            s=ibec_copy(s,p2,500);
            p=ibec_pos('location',s)+10;
            s=ibec_copy(s,p,500);
            p=ibec_pos('"lat"',s)+8;
            s=ibec_copy(s,p,500);
            p2=ibec_pos(',',s)-1;
            slat=ibec_copy(s,1,p2);

            s=ibec_copy(s,p2,500);
            p=ibec_pos('"lng"',s)+8;
            s=ibec_copy(s,p,500);
            slon=ibec_trim(ibec_copy(s,1,15));
            if (sadr<>'')
            then
            begin
              sadr=ibec_UTF8ToAnsiString(sadr);
              sadr=ibec_copy(sadr,1,255);
              try
                update address set address.lon=:slon, address.lat=:slat, address.adr=:sadr where
address.id=:id;
              except
                update address set address.lon=0, address.lat=0, address.adr=:sadr where
address.id=:id;
              end
            end
            else
              update address set address.lon=0, address.lat=0, address.adr='' where
address.id=:id;
          end
        end
        ibec_Progress(i||'  '||sadr);
        ibec_Sleep(2000);
      end
    end
    commit;
  finally
    ibec_CloseConnection(DB);
  end;
end;
```

The example posted above is ready to use, you just need to adjust your database connection data and to supply your own Google cloud services key. In essence, this script will send some data to Google cloud services, fetch the resulting dataset in JSON format, load it and parse the data, so you can collect correct and up-to-date addresses as well as GPS data.

IBEBlock really is a crown jewel! It is a very good reason to buy the product (if you don't have it already).