# Data Comparer Using Cursors

The following example illustrates the use of cursors to compare two tables in different databases.

```
execute ibeblock (
   ProcessInserts boolean = TRUE,
   ProcessUpdates boolean = TRUE,
   ProcessDeletes boolean = TRUE)
returns (
   InsertedRecs integer = 0 comment 'Records inserted',
   UpdatedRecs integer = 0 comment 'Records updated',
   DeletedRecs integer = 0 comment 'Records deleted',
   TotalTime double precision = 0 comment 'Time spent (seconds)')
as
begin
   RecNum = 50000; -- How many records will be inserted into our test table
```

If the databases already exist we will not try to create them. Of course, this approach does not apply to remote databases.

```
  if (not ibec_fileexists('d:\MasterDB.fdb')) then
     ibec_CreateDatabase(__ctFirebird, 'DBName="localhost:d:\MasterDB.fdb";
                                ClientLib="fbclient.dll";
                                User=SYSDBA; Password=masterkey;
PageSize=16384;
                                DefaultCharset=UTF8; SQLDialect=3');
```

CLIENTLIB isn't mandatory if you're using the standard gds32.dll.

```
  if (not ibec_fileexists('d:\SubscriberDB.fdb')) then
     ibec_CreateDatabase(__ctFirebird,
'DBName="localhost:d:\SubscriberDB.fdb";
                                ClientLib="fbclient.dll";
                                User=SYSDBA; Password=masterkey;
PageSize=16384;
                                DefaultCharset=UTF8; SQLDialect=3');
```

Creating two named connections to our databases...

```
  MasterDB =
ibec_CreateConnection(__ctFirebird,'DBName="localhost:d:\MasterDB.FDB";
                                  ClientLib=fbclient.dll;
                                  user=SYSDBA; password=masterkey; names=UTF8;
sqldialect=3');
  SubscriberDB =
ibec_CreateConnection(__ctFirebird,'DBName="localhost:d:\SubscriberDB.FDB";
                                  ClientLib=fbclient.dll;
                                  user=SYSDBA; password=masterkey; names=UTF8;
```

```
sqldialect=3');
```

Now we shall create the IBE$$TEST_DATA table in each database and populate it with some data:

```
  CreateStmt =
     'create table IBE$$TEST_DATA (
       ID integer not null,
       ID2 varchar(20) not null,
       F_INTEGER integer,
       F_VARCHAR varchar(100),
       F_DATE date,
       F_TIME time,
       F_TIMESTAMP timestamp,
       F_NUMERIC numeric(15,2),
       F_BOOL char(1) check (F_BOOL in ('T', 'F')),
       F_BLOB blob sub_type 1,
       F_SEASON varchar(15) check(F_SEASON in ('Spring', 'Summer', 'Autumn',
'Winter')))';
```

IBE$$TEST_DATA will have a primary key consisting of two fields. This is just to demonstrate how to do this when a primary key consists of more than one field.

```
  AlterStmt =
     'alter table IBE$$TEST_DATA add constraint PK_IBE$$TEST_DATA primary
key (ID, ID2)';
```

First we're working with the MasterDB:

```
  ibec_UseConnection(MasterDB);
```

If IBE$$TEST_DATA doesn't exist in the database we must create it:

```
  if (not exists(select rdb$relation_name from rdb$relations where
rdb$relation_name = 'IBE$$TEST_DATA')) then
  begin
```

Creating the table itself...

```
     execute statement :CreateStmt;
```

DDL statements must be committed explicitly:

```
     commit;
```

...and create a primary key:

```
     execute statement :AlterStmt;
     commit;
```

So, we've just created the table. Now we should populate it with data. We will generate some random data for each field, and use an autoincrement for the first primary key field value:

```
     i = 0;
     while (i < RecNum) do
     begin
        fid2    = ibec_randomstring(1,20,65,90);
        fint    = ibec_random2(1, 100000);
        fvarc   = ibec_randomstring(1,100,65,90);
        fdate   = ibec_random2(20000,40000);
        ftime   = ibec_random(0);
        ftimest = ibec_random2(20000,40000) + ibec_random(0);
        fnum    = ibec_random2(1,40000) + ibec_random(0);
        fbool   = ibec_randomval('T','F');
        fblob   = ibec_randomstring(500, 1000, 65, 90);
        fseason = ibec_randomval('Spring'', 'Summer', 'Autumn', 'Winter');
        insert into IBE$$TEST_DATA values (:i, :fid2, :fint, :fvarc, :fdate,
:ftime, :ftimest, :fnum, :fbool, :fblob, :fseason);
        i = i + 1;
```

We will display a progress message after each 500 records inserted. In the SQL Editor it will be displayed on the progress panel above the code editor.

```
        if (ibec_mod(i, 500) = 0) then
        begin
           ibec_progress(i || ' records inserted...');
```

Don't forget to commit!

```
           commit;
        end
     end
```

Once more COMMIT. Maybe there are some uncommited INSERTs...

```
     commit;
  end
```

Let's work with the second connection...

```
  ibec_UseConnection(SubscriberDB);
```

If IBE$$TEST_DATA doesn't exist in the database we must create it:

```
  if (not exists(select rdb$relation_name from rdb$relations where
rdb$relation_name =  ''IBE$$TEST_DATA'')) then
  begin
  execute statement :CreateStmt;
```

Don't forget to commit each DDL statement explicitly!

```
  commit;
  execute statement :AlterStmt;
  commit;
```

The idea is that we fetch the data from the first database and insert it into IBE$$TEST_TABLE in the second database:

```
  ibec_UseConnection(MasterDB);

  i = 0;
  k = 0;
```

FOR ... SELECT will select data from the first database...

```
  for select * from IBE$$TEST_DATA
  into vals
  do
  begin
```

...and we will insert them into the second database:

```
     use SubscriberDB;
     k = k + 1; -- Just a counter...
```

Now we should modify some of the data. Otherwise we'll have nothing to compare

```
     if (ibec_mod(k,100) <> 0) then
```

Each hundredth record will be skipped...

```
     begin
        if (ibec_mod(i,10) = 0) then
```

the 8th field of each tenth record will be changed to NULL...

```
           vals[7] = null;
        if (ibec_mod(i,30) = 0) then
```

...and the 10th field of each 30th record will be modified...

```
           vals[9] = ibec_randomstring(500, 1000, 0, 255);
```

Finally insert a record:

```
        insert into SubscriberDB.IBE$$TEST_DATA values :vals;
        i = i + 1;
```

After each 500 inserted records we will display a progress message. We will also commit after every

500 INSERTs:

```
        if (ibec_mod(i, 500) = 0) then
        begin
           ibec_progress(i || ' records inserted...');
           commit;
        end
     end
  end
```

Once again COMMIT...

```
  ibec_UseConnection(SubscriberDB);
  commit;
```

Now we will insert more data into the second database to provide further discrepancies between the two tables...

```
  i = k + 1;
  while (i < (RecNum + 1000 + 1)) do
  begin
     fid2    = ibec_randomstring(1,20,65,90);
     fint    = ibec_random2(1, 100000);
     fvarc   = ibec_randomstring(1,100,65,90);
     fdate   = ibec_random2(20000,40000);
     ftime   = ibec_random(0);
     ftimest = ibec_random2(20000,40000) + ibec_random(0);
     fnum    = ibec_random2(1,40000) + ibec_random(0);
     fbool   = ibec_randomval('T','F');
     fblob   = ibec_randomstring(500, 1000, 65, 90);
     fseason = ibec_randomval('Spring', 'Summer', 'Autumn', 'Winter');

     insert into IBE$$TEST_DATA values (:i, :fid2, :fint, :fvarc, :fdate,
:ftime, :ftimest, :fnum, :fbool, :fblob, :fseason);

     if (ibec_mod(i, 500) = 0) then
     begin
        ibec_progress(i || ' records inserted...');
        commit;
     end
     i = i + 1;
  end
  commit;
end
```

So, let's begin to compare data. Our goal is make the second IBE$$TEST_DATA a full copy of the first IBE$$TEST_DATA.

First of all we should get the primary key of the reference table:

```
ibec_UseConnection(MasterDB);
```

```
  i = 0;
  for select i.rdb$field_name
  from rdb$relation_constraints rc, rdb$index_segments i, rdb$indices idx
  where (i.rdb$index_name = rc.rdb$index_name) and
        (idx.rdb$index_name = rc.rdb$index_name) and
        (rc.rdb$constraint_type = 'PRIMARY KEY') and
        (rc.rdb$relation_name = 'IBE$$TEST_DATA')
  order by i.rdb$field_position
  into fldname
  do
  begin
     PKFields[i] = ibec_trim(fldname);
     i = i + 1;
  end
```

Now we need to get a list of remaining fields:

```
  SelStmt = 'select rdb$field_name
             from rdb$relation_fields
             where (rdb$relation_name = ''IBE$$TEST_DATA'')';
```

Here we add a condition to exclude primary key fields from the SELECT result:

```
  i = 0;
  HighDim = ibec_high(PKFields);
  for i = 0 to HighDim do
     SelStmt = SelStmt || ' and (rdb$field_name <> ' ||
ibec_QuotedStr(PKFields[i], '''') || ')';
```

We need the natural order of the fields...

```
  SelStmt = SelStmt || ' order by rdb$field_position';
```

Finally execute the SELECT statement just created and get an array of all non-PK fields:

```
  i = 0;
  for execute statement :SelStmt
  into :s
  do
  begin
```

Trim spaces, we don't need them...

```
     NonPKFields[i] = ibec_trim(:s);
     i = i + 1;
  end
```

Let's compose necessary statements:

SelStmt will be used to retrieve data UpdStmt will be used to update the second table if two records differ:

```
  SelStmt = '';
  UpdStmt = 'update ibe$$test_data set ';
  WhereClause = ' where ';

  HighDim = ibec_high(NonPKFields);
  for i = 0 to HighDim do
  begin
     SelStmt = SelStmt || NonPKFields[i];
     SelStmt = SelStmt || ', ';
     UpdStmt = UpdStmt || ibec_chr(13) || NonPKFields[i] || ' = :' ||
NonPKFields[i];
     if (i < HighDim) then
        UpdStmt = UpdStmt || ', ';
  end
```

Here we compose a WHERE clause with primary key fields: WHERE (PK_FIELD1 = :PK_FIELD1) AND (PK_FIELD2 = :PK_FIELD2) AND ...

```
  HighDim = ibec_high(PKFields);
  for i = 0 to HighDim do
  begin
     SelStmt = SelStmt || ibec_trim(PKFields[i]);
     WhereClause = WhereClause || '(' || ibec_trim(PKFields[i]) || ' = :' ||
ibec_trim(PKFields[i]) || ')';
     if (i < HighDim) then
     begin
        SelStmt = SelStmt || ', ';
        WhereClause = WhereClause || ' and ';
     end
  end

  SelStmt = 'select ' || SelStmt || ' from IBE$$TEST_DATA order by ';

  for i = 0 to HighDim do
  begin
     SelStmt = SelStmt || ibec_trim(PKFields[i]);
     if (i < HighDim) then
        SelStmt = SelStmt || ', ';
  end

  PKFieldCount = ibec_high(PKFields)+1;
  PKFieldIndex = ibec_high(NonPKFields)+1;

  StartTime = ibec_gettickcount(); -- Note the time...
```

```
  MasterCR = ibec_cr_OpenCursor(MasterDB, SelStmt);
  SubscriberCR = ibec_cr_OpenCursor(SubscriberDB, SelStmt);
```

Compose the INSERT statement:

```
  InsFields = ''; InsValues = '';
  FldCount = ibec_cr_FieldCount]](SubscriberCR);
  for i = 0 to (FldCount-1) do
  begin
     FldName = ibec_Trim(ibec_cr_FieldName(SubscriberCR, i));
     InsFields = InsFields || FldName;
     InsValues = InsValues || '':' || FldName;
     if (i < (FldCount-1)) then
     begin
        InsFields = InsFields || ', ';
        InsValues = InsValues || ', ';
     end
  end
  InsStmt = 'insert into ibe$$test_data (' || InsFields || ') values (' ||
InsValues || ')';

  ibec_UseConnection(SubscriberDB);

  while (not (ibec_cr_Eof]](MasterCR) and ibec_cr_Eof(SubscriberCR))) do
  begin
     CompResult = 0;
     if (ibec_cr_Eof(MasterCR)) then
        CompResult = 1;
     else if (ibec_cr_Eof(SubscriberCR)) then
        CompResult = -1;
     else
     begin
        ibec_cr_Fetch(MasterCR, MasterPK, PKFieldIndex, PKFieldCount);
        ibec_cr_Fetch(SubscriberCR, SubscriberPK, PKFieldIndex,
PKFieldCount);
        CompResult = ibec_CmpRecords2(MasterPK, SubscriberPK);
     end

     if (ProcessUpdates and (CompResult = 0)) then
     begin
        ibec_cr_Fetch(MasterCR, MasterVals, 0, PKFieldIndex);
        ibec_cr_Fetch(SubscriberCR, SubscriberVals, 0, PKFieldIndex);
        CompResult = ibec_CmpRecords(MasterVals, SubscriberVals);
        if (CompResult <> -1) then
        begin
           UpdatedRecs = UpdatedRecs + 1;
           ibec_progress('Record must be updated...');
           ibec_cr_Fetch(MasterCR, MasterVals, 0, null);
           execute statement :UpdStmt || WhereClause values :MasterVals;
        end

        ibec_cr_Next(MasterCR);
        ibec_cr_Next(SubscriberCR);
```

```
    end
    else if (ProcessInserts and (CompResult < 0)) then
    begin
```

Redundant master record found. Insert it into the subscriber:

```
       InsertedRecs = InsertedRecs + 1;
       ibec_progress('Record must be inserted...');
       ibec_cr_Fetch(MasterCR, MasterVals, 0, null);
       execute statement :InsStmt values :MasterVals;
       ibec_cr_Next(MasterCR);
    end
    else if (ProcessDeletes and (CompResult > 0)) then
    begin
```

Redundant subscriber record found. Delete it.

```
       DeletedRecs = DeletedRecs + 1;
       ibec_progress('Record must be deleted...');
       ibec_cr_Fetch(SubscriberCR, SubscriberPK, PKFieldIndex,
PKFieldCount);
       execute statement 'delete from ibe$$test_data ' || WhereClause
values :SubscriberPK;
      ibec_cr_Next(SubscriberCR);
    end;
  end

  ibec_cr_CloseCursor(MasterCR);
  ibec_cr_CloseCursor(SubscriberCR);

  commit;
```

Done. Close both connections:

```
  close connection MasterDB;
  close connection SubscriberDB;
```

Let's count the elapsed time...

```
  EndTime = ibec_gettickcount();
  TotalTime = (EndTime - StartTime) / 1000;
  suspend;
end
```

Full Example: Data Comparer Using Cursors

From:
https://www.ibexpert.com/docu/ - **IBExpert**

Permanent link:
**https://www.ibexpert.com/docu/doku.php?id=06-ibexpert-ibeblock-examples:data-comparer-using-cursors**

Last update: **2023/07/06 17:22**