Copy Table Example

```
 execute ibeblock (
 -- Don't change names of following parameters! ------------------------------
--------------------
   SrcDBConnStr variant comment 'Source DB connection string',
   SrcDBUserName variant = 'SYSDBA' comment 'Source DB user name',
   SrcDBPassword variant = 'masterkey' comment 'Source DB password',
   SrcDBCharset variant = 'NONE' comment 'Source DB connection charset',
   SrcDBClientLib variant = 'gds32.dll' comment 'Source DB client library
name',
   DestDBConnStr variant comment 'Destination DB connection string',
   DestDBUserName variant = 'SYSDBA' comment 'Destination DB user name',
   DestDBPassword variant = 'masterkey' comment 'Destination DB password',
   DestDBCharset variant = 'NONE' comment 'Destination DB connection
charset',
   DestDBClientLib variant = 'gds32.dll' comment 'Destination DB client
library name',
   SrcObjectName variant = '' comment 'Table name to be copied',
   DestObjectName variant = '' comment 'Destination table name, leave empty
if no changes need',
   DebugMode boolean = TRUE,
 ----------------------------------------------------------------------------
--------------------
   CopyDomains boolean = TRUE comment 'Copy domains',
   CopyTriggers boolean = TRUE comment 'Copy table triggers',
   CopyPrimaryKey boolean = TRUE comment 'Copy primary key',
 --  CopyForeignKeys boolean = FALSE comment 'Copy foreign keys',
   CopyGenerators boolean = TRUE comment 'Copy generators used within table
triggers',
   CopyData boolean = TRUE comment 'Copy table data',
   CopyIndices boolean = TRUE comment 'Copy table indices',
   DropTableIfExists boolean = FALSE comment 'Try to drop table if the one
exists in the destination database')
 as
 begin
   Time1 = ibec_GetTickCount();

   CRLF = ibec_CRLF();
   BS = ibec_Chr(8);
   Success = BS + ' Successfull.';
   Failed = BS + ' FAILED!';

   if (DebugMode) then
   begin
     SrcDBConnStr = 'LOCALHOST/3060:D:\FB2_DATA\IBEHELP.FBA';
     SrcDBCharset = 'WIN1251';
     SrcDBClientLib = 'C:\Program Files\Firebird\bin\fbclient.dll';
```

```
    --DestDBConnStr = 'AVX-MAIN:D:\FB2_DATA\FORMTEST.FDB';
    DestDBConnStr = 'LOCALHOST/3060:D:\FB2_DATA\IBEHELP.FBA';
    DestDBCharset = 'WIN1251';
    DestDBClientLib = 'C:\Program Files\Firebird\bin\fbclient.dll';

    SrcObjectName = 'HELP_ITEMS';
    DestObjectName = 'HELP_ITEMS33';
    DropTableIfExists = TRUE;
  end;

  SrcTableName = SrcObjectName;
  DestTableName = DestObjectName;


  SrcDBParams = 'DBName=' + SrcDBConnStr + ';' +
                'User=' + SrcDBUserName + ';' +
                'Password=' + SrcDBPassword + ';' +
                'Names=' + SrcDBCharset + ';' +
                'ClientLib=' + SrcDBClientLib;

  DestDBParams = 'DBName=' + DestDBConnStr + ';' +
                 'User=' + DestDBUserName + ';' +
                 'Password=' + DestDBPassword + ';' +
                 'Names=' + DestDBCharset + ';' +
                 'ClientLib=' + DestDBClientLib;

  try
    try
      ibec_Progress('Connecting to ' + SrcDBConnStr + '...');
      SrcDB = ibec_CreateConnection(__ctFirebird, SrcDBParams);
      ibec_Progress(Success);
      SrcDBSQLDialect = ibec_GetConnectionProp(SrcDB, 'DBSQLDialect');
    except
      ibec_Progress(Failed);
      raise;
      Exit;
    end;

  try
    ibec_Progress('Connecting to ' + DestDBConnStr + '...');
    DestDB = ibec_CreateConnection(__ctFirebird, DestDBParams);
    ibec_Progress(Success);
    DestDBSQLDialect = ibec_GetConnectionProp(DestDB, 'DBSQLDialect');
  except
    ibec_Progress(Failed);
    raise;
    Exit;
  end;

  ibec_UseConnection(SrcDB);
```

```
   select rdb$relation_name, rdb$system_flag, rdb$external_file,
rdb$description
          from rdb$relations
          where (rdb$relation_name = :SrcTableName) and (rdb$view_blr is
null)
          into :SrcTableData;

   if (SrcTableData['RDB$RELATION_NAME'] is null) then
     exception cant_find_table 'There is no such table (' + :SrcTableName +
') in the source database.';
   IsSys = SrcTableData['RDB$SYSTEM_FLAG'] = 1;
   if (IsSys) then
     exception cant_copy_system_table 'Cannot copy a system table.';


   if ((DestTableName is null) or (DestTableName = ''))  then
     DestTableName = SrcTableName;

   DestTableNameFmt = ibec_IIF(DestDBSQLDialect = 3,
ibec_QuotedStr(:DestTableName, '"'), ibec_AnsiUpperCase(:DestTableName));
   SrcTableNameFmt = ibec_IIF(SrcDBSQLDialect = 3,
ibec_QuotedStr(:SrcTableName, '"'), ibec_AnsiUpperCase(:SrcTableName));

   ibec_UseConnection(DestDB);

   if (exists(select rdb$relation_name from rdb$relations where
rdb$relation_name = :DestTableName)) then
   begin
     if (DropTableIfExists) then
     begin
       DropStmt = 'DROP TABLE ' + DestTableNameFmt;

       try
         ibec_Progress('Dropping table ' + DestTableNameFmt + '...');
         execute statement :DropStmt;
         commit;
         ibec_Progress(Success);
       except
         ibec_Progress(Failed);
         rollback;
         raise;
       end;
     end
     else
       exception table_exists_already 'Table "' + DestTableName + '" exists
in the destination database already.';
   end

   ibec_UseConnection(SrcDB);

   select rdb$field_name
```

```
          from rdb$relation_fields
          where (rdb$relation_name = 'RDB$FIELDS') and
                (rdb$field_name = 'RDB$FIELD_PRECISION')
          into :bPrecision;
   bPrecision = ibec_IIF(:bPrecision is NULL, FALSE, TRUE);

   SelStmt = 'select rf.rdb$field_name as fld_name,' +
                    'rf.rdb$field_source as fld_domain,' +
                    'rf.rdb$null_flag as fld_null_flag,' +
                    'rf.rdb$default_source as fld_default,' +
                    'rf.rdb$description as fld_description,' +
                    'f.rdb$field_type as dom_type,' +
                    'f.rdb$field_length as dom_length,' +
                    'f.rdb$field_sub_type as dom_subtype,' +
                    'f.rdb$field_scale as dom_scale,' +
                    'f.rdb$null_flag as dom_null_flag,' +
                    'f.rdb$character_length as dom_charlen,' +
                    'f.rdb$segment_length as dom_seglen,' +
                    'f.rdb$system_flag as dom_system_flag,' +
                    'f.rdb$computed_source as dom_computedby,' +
                    'f.rdb$default_source as dom_default,' +
                    'f.rdb$dimensions as dom_dims,' +
                    'f.rdb$description as dom_description,' +
                    'ch.rdb$character_set_name as dom_charset,' +
                    'ch.rdb$bytes_per_character as charset_bytes,' +
                    'dco.rdb$collation_name as dom_collation,' +
                    'fco.rdb$collation_name as fld_collation';
   if (bPrecision) then
     SelStmt = SelStmt + ', f.rdb$field_precision as dom_precision';

   SelStmt = SelStmt + CRLF +
             'from rdb$relation_fields rf ' + CRLF +
             'left join rdb$fields f on rf.rdb$field_source =
f.rdb$field_name' + CRLF +
             'left join rdb$character_sets ch on f.rdb$character_set_id =
ch.rdb$character_set_id' + CRLF +
             'left join rdb$collations dco on ((f.rdb$collation_id =
dco.rdb$collation_id) and (f.rdb$character_set_id =
dco.rdb$character_set_id))' + CRLF +
             'left join rdb$collations fco on ((rf.rdb$collation_id =
fco.rdb$collation_id) and (f.rdb$character_set_id =
fco.rdb$character_set_id))' + CRLF +
             'where rf.rdb$relation_name = ' + ibec_QuotedStr(:SrcTableName,
'''') + CRLF +
             'order by rf.rdb$field_position';

   ibec_Progress('Collecting fields info...');
   i = 0;
   iUserDomainCount = 0;
   for execute statement SelStmt into :FldData
   do
   begin
     s = ibec_Trim(FldData['FLD_DOMAIN']);
     aDomains[i] = ibec_IIF(ibec_Copy(s, 1, 4) = 'RDB$', null, s);
     if (aDomains[i] is not null) then
       iUserDomainCount = iUserDomainCount + 1;

     aFields[i] = ibec_Trim(FldData['FLD_NAME']);

     sType = ibec_IBTypeToStr(FldData['DOM_TYPE'],
                              FldData['DOM_SUBTYPE'],
                              FldData['DOM_LENGTH'],
                              FldData['DOM_SCALE'],
                              FldData['DOM_SEGLEN'],
                              FldData['DOM_CHARLEN'],
                              FldData['DOM_PRECISION'],
                              DestDBSQLDialect);
     aTypes[i] = sType;

     aFieldsNotNull[i] = ibec_IIF(FldData['FLD_NULL_FLAG'] = 1, ' NOT NULL',
'');
     aFieldsDefault[i] = ibec_IIF(FldData['FLD_DEFAULT'] is null, '', ' ' +
ibec_Trim(FldData['FLD_DEFAULT']));
     aFieldsComment[i] = FldData['FLD_DESCRIPTION'];
     aFieldsCharset[i] = ibec_IIF(FldData['DOM_CHARSET'] is null, '',
ibec_Trim(FldData['DOM_CHARSET']));
     aFieldsCollate[i] = ibec_IIF(FldData['FLD_COLLATION'] is null, '',
ibec_Trim(FldData['FLD_COLLATION']));

     aDomainsComputedBy[i] = FldData['DOM_COMPUTEDBY'];
     i = i + 1;
   end

   ibec_UseConnection(DestDB);
   DomainsAreOK = TRUE;
   if (CopyDomains and (iUserDomainCount > 0)) then
   begin
     ibec_Progress('Creating domains...');
     foreach (aDomains as Dom key DomIdx skip nulls) do
     begin
       if (exists(select rdb$field_name from rdb$fields where rdb$field_name
= :Dom)) then
         Continue;
       CreateStmt = 'CREATE DOMAIN ' +
                    ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(:Dom,
'"'), ibec_AnsiUpperCase(:Dom)) +
                    ' AS ' +  sType;
       try
         execute statement :CreateStmt;
         commit;
       except
```

```
        DomainsAreOK = FALSE;
        rollback;
      end;
    end;
  end

  FieldsList = '';

  CreateStmt = 'CREATE TABLE ' + DestTableNameFmt;
  foreach (aFields as FldName index FldKey skip nulls) do
  begin
    sType = '';
    if (FieldsList <> '') then
      FieldsList .= ',' + CRLF;
    FldNameFmt = ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(:FldName,
'"'), ibec_AnsiUpperCase(:FldName));
    if (DomainsAreOK and (aDomains[FldKey] is not null)) then
      FieldsList .= FldNameFmt + ' ' + aDomains[FldKey];
    else
    FieldsList .= FldNameFmt + ' ' + aTypes[FldKey];
    if ((aDomains[FldKey] is null) and (aFieldsCharset[FldKey] <> '')) then
      FieldsList .= ' CHARACTER SET ' + aFieldsCharset[FldKey];
    FieldsList .= aFieldsDefault[FldKey] + aFieldsNotNull[FldKey];
    if (aFieldsCollate[FldKey] <> '') then
      FieldsList .= ' COLLATE ' + aFieldsCollate[FldKey];
  end
  CreateStmt .= ' (' + CRLF + FieldsList + ')';

  ibec_UseConnection(DestDB);
  try
    ibec_Progress('Creating table ' + DestTableNameFmt + '...');
    execute statement :CreateStmt;
    commit;
    ibec_Progress(Success);

    TblName = ibec_IIF(DestDBSQLDialect = 3, :DestTableName,
ibec_AnsiUpperCase(:DestTableName));
    foreach (aFieldsComment as FldComment key FldKey skip nulls) do
    begin
      FldName = aFields[FldKey];
      update rdb$relation_fields set rdb$description = :FldComment
             where (rdb$relation_name = :TblName) and (rdb$field_name =
:FldName);
    end;
    commit;
  except
    ibec_Progress(Failed);
    rollback;
  end;
```

```
   -------------------------------------------------------------
   -- TRANSFER TABLE DATA ---------------------------------------
   -------------------------------------------------------------
   if (CopyData) then
   begin
     sFields = '';
     sValues = '';
     foreach (aFields as FldName key FldKey) do
     begin
       if (aDomainsComputedBy[FldKey] is null) then
       begin
         if (sFields <> '') then
         begin
           sFields .= ', ';
           sValues .= ', ';
         end;
         FldNameFmt = ibec_IIF(DestDBSQLDialect = 3,
ibec_QuotedStr(:FldName, '"'), ibec_AnsiUpperCase(:FldName));
         sFields .= FldNameFmt;
         sValues .= ':' + FldNameFmt;
       end;
     end;

     SelectStmt = 'SELECT ' + sFields + ' FROM ' + SrcTableNameFmt;
     InsertStmt = 'INSERT INTO ' + DestTableNameFmt + ' (' + sFields + ')
VALUES (' + sValues + ')';

     ibec_UseConnection(SrcDB);
     i = 0;
     ibec_Progress('Copying table data...');
     for execute statement :SelectStmt into :Data
     do
     begin
       ibec_UseConnection(DestDB);
       execute statement :InsertStmt values :Data;
       i = i + 1;
       if (ibec_mod(i, 500) = 0) then
       begin
         commit;
         ibec_Progress('    ' + ibec_cast(i, __typeString) + ' records
copied...');
       end;
     end;
     ibec_Progress('Totally ' + ibec_cast(i, __typeString) + ' records
copied.');
     ibec_UseConnection(DestDB);
     commit;
   end;

   if (CopyTriggers or CopyPrimaryKey or CopyGenerators) then
   begin
```

```
     ibec_UseConnection(SrcDB);
     TblName = ibec_IIF(SrcDBSQLDialect = 3, :SrcTableName,
ibec_AnsiUpperCase(:SrcTableName));
     i = 0;
     ibec_Progress('Collecting triggers info...');
     for select T.RDB$TRIGGER_NAME, T.RDB$TRIGGER_TYPE,
T.RDB$TRIGGER_SEQUENCE,
                T.RDB$TRIGGER_INACTIVE, T.RDB$TRIGGER_SOURCE
         from RDB$TRIGGERS T
         left join RDB$CHECK_CONSTRAINTS C on C.RDB$TRIGGER_NAME =
T.RDB$TRIGGER_NAME
         where ((T.RDB$SYSTEM_FLAG = 0) or (T.RDB$SYSTEM_FLAG is null)) and
               (C.rdb$trigger_name is null) and (T.RDB$RELATION_NAME =
:TblName)
         order by T.RDB$TRIGGER_NAME
         into :TrgData
     do
     begin
       aTriggerNames[i] = ibec_Trim(TrgData['RDB$TRIGGER_NAME']);
       aTriggerTypes[i] =
ibec_IBTriggerTypeToStr(TrgData['RDB$TRIGGER_TYPE']);
       aTriggerPositions[i] = TrgData['RDB$TRIGGER_SEQUENCE'];
       aTriggerInactives[i] = ibec_IIF(TrgData['RDB$TRIGGER_INACTIVE'] = 1,
'INACTIVE', 'ACTIVE');
       aTriggerSources[i] = TrgData['RDB$TRIGGER_SOURCE'];
       i = i + 1;
     end;

     select rc.rdb$constraint_name,
            rc.rdb$index_name
     from rdb$relation_constraints rc
     where (rc.rdb$constraint_type = 'PRIMARY KEY') and
(rc.rdb$relation_name = :TblName)
     into :PKData;

     if (PKData is not null) then
     begin
       i = 0;
       PKIdxName = ibec_Trim(PKData[1]);
       for select rdb$field_name
           from rdb$index_segments
           where rdb$index_name = :PKIdxName
           order by rdb$field_position
           into :PKField
       do
       begin
         PKFields[i] = ibec_Trim(:PKField);
         i = i + 1;
       end
     end;
     -----------------------------------------------------
     -- COLLECTING GENERATOR NAMES USED WITHIN TRIGGERS
     -----------------------------------------------------

     i = 0;
     ibec_Progress('Searching trigger bodies for used generators...');
     foreach (aTriggerNames as TrgName key TrgKey skip nulls) do
     begin
       TrgSrc = aTriggerSources[TrgKey];
       TrgNameFmt = ibec_IIF(SrcDBSQLDialect = 3, ibec_QuotedStr(:TrgName,
'"'), ibec_AnsiUpperCase(:TrgName));
       TrgDDL = 'CREATE TRIGGER ' + TrgNameFmt + ' FOR ' + SrcTableNameFmt +
CRLF +
                aTriggerTypes[TrgKey] + ' POSITION ' +
ibec_Cast(aTriggerPositions[TrgKey], __typeString) + CRLF + TrgSrc;
       PSQLParser = ibec_psql_Parse(TrgDDL, SrcDBSqlDialect, __svUnknown);
       try
         if (ibec_psql_ErrorCount(PSQLParser) = 0) then
         begin
           iCount = ibec_psql_UsedObjects(PSQLParser, ObjNames, ObjTypes);
           if (iCount > 0) then
           begin
             foreach (ObjNames as ObjName key ObjKey skip nulls) do
               if (ObjTypes[ObjKey] = __dboGenerator) then
                 if (ibec_IndexOfValue(Generators, ObjName) is null) then
                 begin
                   Generators[i] = ObjName;
                   i = i + 1;
                 end;
           end;
         end;
       finally
         ibec_psql_Free(PSQLParser);
       end;
     end;

     -------------------------------------------------------
     -- CREATING GENERATORS AND SETTING THEIR VALUES
     -------------------------------------------------------

     ibec_Progress('Creating and initting generators...');
     foreach (Generators as GenName key GenKey skip nulls) do
     begin
       ibec_UseConnection(DestDB);
       if (exists(select rdb$generator_name from rdb$generators where
rdb$generator_name = :GenName)) then
         Continue;

       ibec_UseConnection(SrcDB);
       GenNameFmt = ibec_IIF(SrcDBSQLDialect = 3, ibec_QuotedStr(:GenName,
'"'), ibec_AnsiUpperCase(:GenName));
```

```
       GetGenValueStmt = 'SELECT GEN_ID(' + GenNameFmt + ', 0) FROM
RDB$DATABASE';
       execute statement GetGenValueStmt into :GenValue;

       GenNameFmt = ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(:GenName,
'"'), ibec_AnsiUpperCase(:GenName));
       CreateGenStmt = 'CREATE GENERATOR ' + GenNameFmt;
       SetGenStmt = 'SET GENERATOR ' + GenNameFmt + ' TO ' +
ibec_Cast(:GenValue, __typeString);

       ibec_UseConnection(DestDB);
       try
         ibec_Progress('    ' + GenNameFmt + '...');
         execute statement CreateGenStmt;
         commit;
         execute statement SetGenStmt;
         commit;
         ibec_Progress(Success);
       except
         ibec_Progress(Failed);
         rollback;
       end;
     end;
   end;

   if (CopyTriggers) then
   begin
     ibec_UseConnection(DestDb);
     ibec_Progress('Creating triggers...');
     foreach (aTriggerNames as TrgName key TrgKey skip nulls) do
     begin
       if (SrcTableName <> DestTableName) then
         TrgName = ibec_preg_Replace('(?i)' + SrcTableName, DestTableName,
TrgName);
       TrgNameFmt = ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(:TrgName,
'"'), ibec_AnsiUpperCase(:TrgName));
       CreateTrgStmt = 'CREATE TRIGGER ' + TrgNameFmt + ' FOR ' +
DestTableNameFmt + CRLF +
                       aTriggerInactives[TrgKey] + ' ' +
aTriggerTypes[TrgKey] + ' POSITION ' + ibec_Cast(aTriggerPositions[TrgKey],
__typeString) + CRLF +
                      aTriggerSources[TrgKey];

       WasError = FALSE;
       try
         ibec_Progress('    ' + TrgNameFmt + '...');
         execute statement :CreateTrgStmt;
         commit;
         ibec_Progress(BS + ' Successfull.');
       except
```

```
         ibec_Progress(BS + ' FAILED!');
         WasError = TRUE;
         rollback;
       end;

       if (WasError) then
       begin
         ibec_Progress('    Attempt to create trigger ' + TrgNameFmt + '
with commented body...');
         PSQLParser = ibec_psql_Parse(CreateTrgStmt, DestDBSqlDialect,
__svUnknown);
         try
           CreateTrgStmt = ibec_psql_CommentBody(PSQLParser);
         finally
           ibec_psql_Free(PSQLParser);
         end;
         try
           execute statement :CreateTrgStmt;
           ibec_Progress(BS + ' Successfull.');
           commit;
         except
           ibec_Progress('    Failed.');
           rollback;
         end;
       end;
     end;
   end;

   if (CopyPrimaryKey) then
   begin
     ibec_UseConnection(SrcDB);
     select rc.rdb$constraint_name, rc.rdb$index_name
            from rdb$relation_constraints rc
            where (rc.rdb$constraint_type = 'PRIMARY KEY') and
(rc.rdb$relation_name = :SrcTableName)
            into :PKData;
     if (PKData is not null) then
     begin
       PKIdxName = ibec_Trim(PKData[1]);
       sFields = '';
       for select rdb$field_name
           from rdb$index_segments
           where rdb$index_name = :PKIdxName
           order by rdb$field_position
           into :PKFields
       do
       begin
         if (sFields <> '') then
           sFields .= ', ';
         FldName = ibec_Trim(PKFields[0]);
         FldNameFmt = ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(FldName,
'"'), ibec_AnsiUpperCase(FldName));
         sFields .= FldNameFmt;
       end;
       PKName = ibec_Trim(PKData[0]);

       ibec_UseConnection(DestDB);
       PKNameBase = 'PK_' + DestTableName + '_';
       PKNameSuff = 0;
       PKExists = 1;
       while (PKExists is not null) do
       begin
         PKNameSuff = PKNameSuff + 1;
         PKName = PKNameBase + ibec_Cast(PKNameSuff, __typeString);
         PKExists = null;
         select 1 from rdb$relation_constraints rc
              where (rc.rdb$constraint_type = 'PRIMARY KEY') and
(rc.rdb$constraint_name = :PKName)
              into :PKExists;
       end;

       PKNameFmt = ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(PKName,
'"'), ibec_AnsiUpperCase(PKName));
       AlterStmt = 'ALTER TABLE ' + DestTableNameFmt + ' ADD CONSTRAINT ' +
PKNameFmt + ' PRIMARY KEY (' + sFields + ')';

       ibec_UseConnection(DestDB);
       try
         ibec_Progress('Creating primary key ' + PKNameFmt + '...');
         execute statement :AlterStmt;
         commit;
         ibec_Progress(Success);
       except
         ibec_Progress(Failed);
         rollback;
       end;
     end;
   end;

   if (CopyIndices) then
   begin
     ibec_Progress('Creating indices...');
     ibec_UseConnection(SrcDB);
     for select i.rdb$index_name, i.rdb$unique_flag, i.rdb$index_inactive,
i.rdb$index_type,
                i.rdb$expression_source, i.rdb$description
         from rdb$indices i
         left join rdb$relation_constraints rc on i.rdb$index_name =
rc.rdb$index_name
         where ((i.rdb$system_flag = 0) or (i.rdb$system_flag is null)) and
(rc.rdb$constraint_name is null)
```

```
               and i.rdb$relation_name = :SrcTableName
         into :IdxData
     do
     begin
       IdxName = ibec_Trim(IdxData[0]);
       IdxNameFmt = ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(IdxName,
'"'), ibec_AnsiUpperCase(IdxName));
       IdxUnique = ibec_IIF((IdxData[1] is null) or (IdxData[1] = 0), '',
'UNIQUE ');
       IdxActive = ibec_IIF((IdxData[2] is null) or (IdxData[1] = 0), '',
'INACTIVE ');
       IdxType = ibec_IIF((IdxData[3] is null) or (IdxData[1] = 0), '',
'DESCENDING ');
       IdxExpression = IdxData[4];
       IdxDescription = IdxData[5];

       sFields = '';
       for select rdb$field_name
           from rdb$index_segments
           where rdb$index_name = :IdxName
           order by rdb$field_position
           into :IdxFields
       do
       begin
         if (sFields <> '') then
           sFields .= ', ';
         FldName = ibec_Trim(IdxFields[0]);
         FldNameFmt = ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(FldName,
'"'), ibec_AnsiUpperCase(FldName));
         sFields .= FldNameFmt;
       end;

       ibec_UseConnection(DestDB);
       IDXExists = null;
       select 1 from rdb$indices where rdb$index_name = :IdxName into
:IDXExists;
       if (IDXExists is not null) then
       begin
         IDXNameBase = 'IDX_' + DestTableName + '_';
         IDXNameSuff = 0;
         IDXExists = 1;
         while (IDXExists is not null) do
         begin
           IDXNameSuff = IDXNameSuff + 1;
           IdxName = IDXNameBase + ibec_Cast(IDXNameSuff, __typeString);
           IDXExists = null;
           select 1 from rdb$indices where rdb$index_name = :IdxName into
:IDXExists;
         end;
         IdxNameFmt = ibec_IIF(DestDBSQLDialect = 3, ibec_QuotedStr(IdxName,
'"'), ibec_AnsiUpperCase(IdxName));
```

```
       end;

       CreateIndexStmt = 'CREATE ' + IdxUnique + IdxType + 'INDEX ' +
IdxNameFmt + ' ON ' +
                         DestTableNameFmt;
       if (IdxExpression is not null) then
         CreateIndexStmt .= ' COMPUTED BY (' + IdxExpression + ')';
       else
         CreateIndexStmt .= ' (' + sFields + ')';

       ibec_UseConnection(DestDB);
       try
         ibec_Progress('    ' + IdxNameFmt + '...');
         execute statement :CreateIndexStmt;
         commit;
         if (IdxActive <> '') then
         begin
           ibec_Progress(BS + ' Making inactive...');
           execute statement 'ALTER INDEX ' || IdxNameFmt || ' INACTIVE';
           commit;
         end;
         ibec_Progress(Success);
       except
         ibec_Progress(Failed);
         rollback;
       end;
       ibec_UseConnection(SrcDB);
     end;
   end;

 finally
   if (SrcDB is not null) then
   begin
     ibec_Progress('Closing connection to ' + SrcDBConnStr + '...');
     ibec_CloseConnection(SrcDB);
   end;
   if (DestDB is not null) then
   begin
     ibec_Progress('Closing connection to ' + DestDBConnStr + '...');
     ibec_CloseConnection(DestDB);
   end;
   Time2 = ibec_GetTickCount();
   sTime = ibec_div((Time2 - Time1), 1000) || '.' ||ibec_mod((Time2 -
Time1), 1000);
   ibec_Progress('Finished.');
   ibec_Progress('Total time spent: ' || sTime || ' seconds');
   ibec_Progress('That''s all, folks!');
 end;
end
```