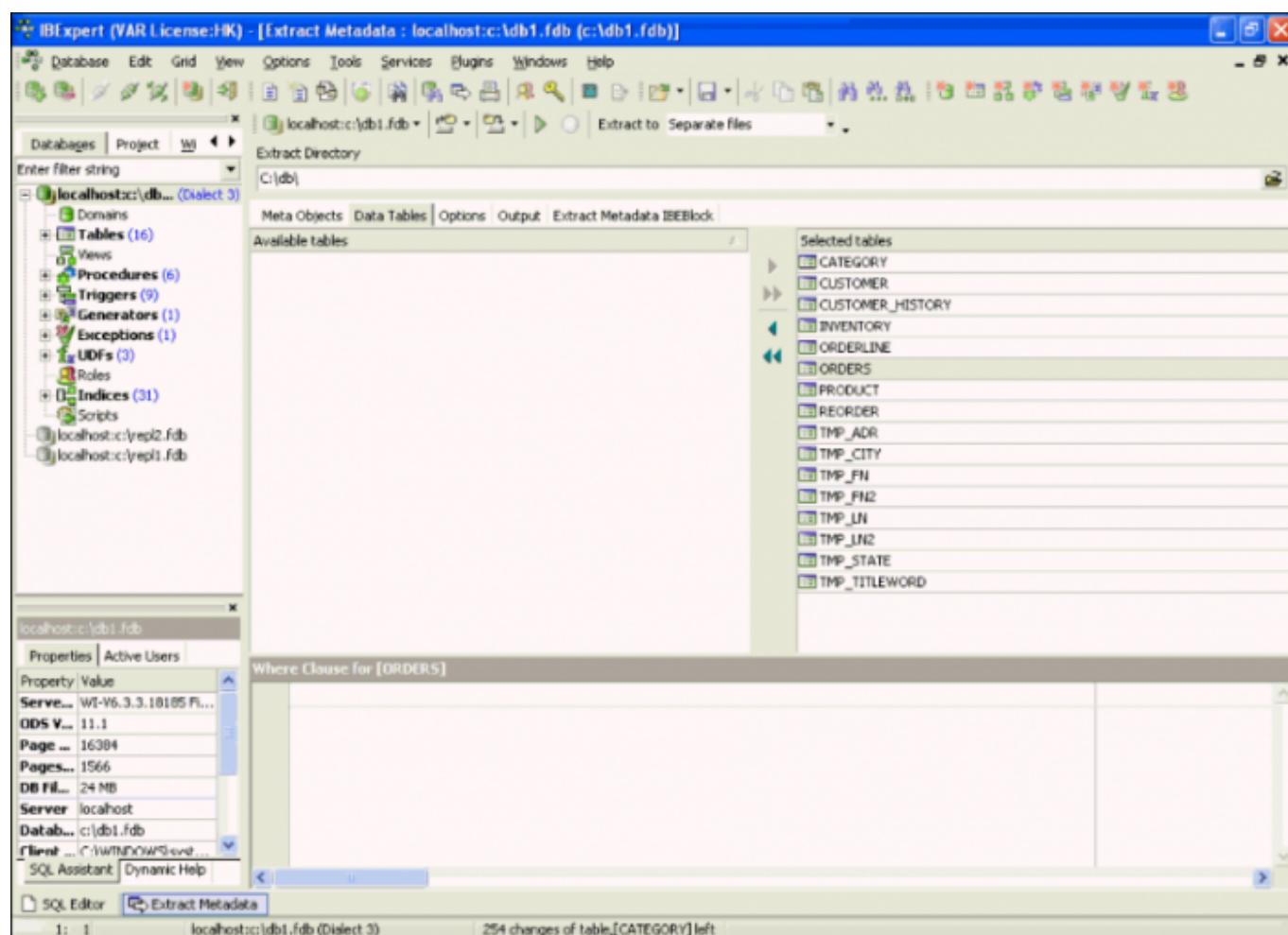


Using Extract Metadata to repair databases

You probably already use the [IBExpert Services](#) menu items, [Backup Database](#), [Restore Database](#), or the [HK-Software Services Control Center](#) to automate your backup and restore. But what happens if you encounter technical problems? For example, the backup is not working properly. You've tried everything to solve it; you've tried to validate the database with GFIX or the IBExpert Services menu item, [Database Validation](#), the API for GFIX operations. And you still have a problem. IBExpert offers you not just the possibility to backup a database but also to extract a database's [metadata](#) and [data](#). IBExpert can [extract metadata](#) and all database data or a selection thereof.

After analyzing your backup log to detect the source of the problem, you can use *Extract Metadata* to separate files, for example, extracting all your objects or all those you know to be undamaged. Then, on the second page, [Data Tables](#) add all data, or limit the data to be extracted by adding a `WHERE` clause:

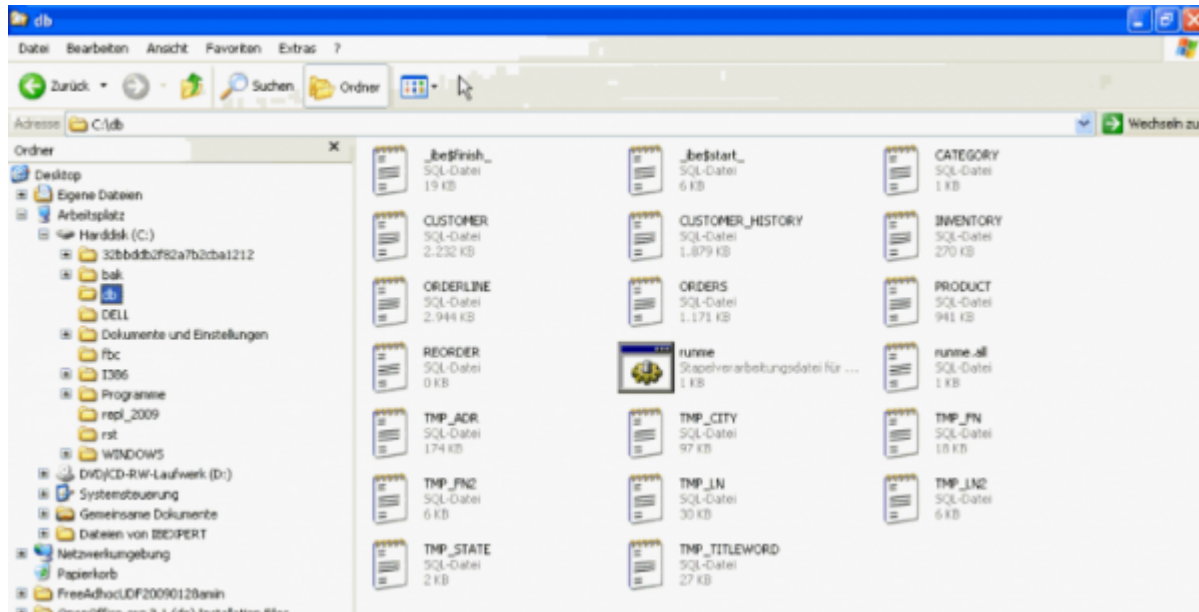


The extract metadata job can be defined as finely as you wish. There is a large range of further options on the [Options](#) page, for example, the file size of the metadata text files can be limited. Some of the options are for compatibility reasons, e.g. use sequence instead of generator, use create or alter for procedures and triggers and so on. And it is even possible to extract the blob data.

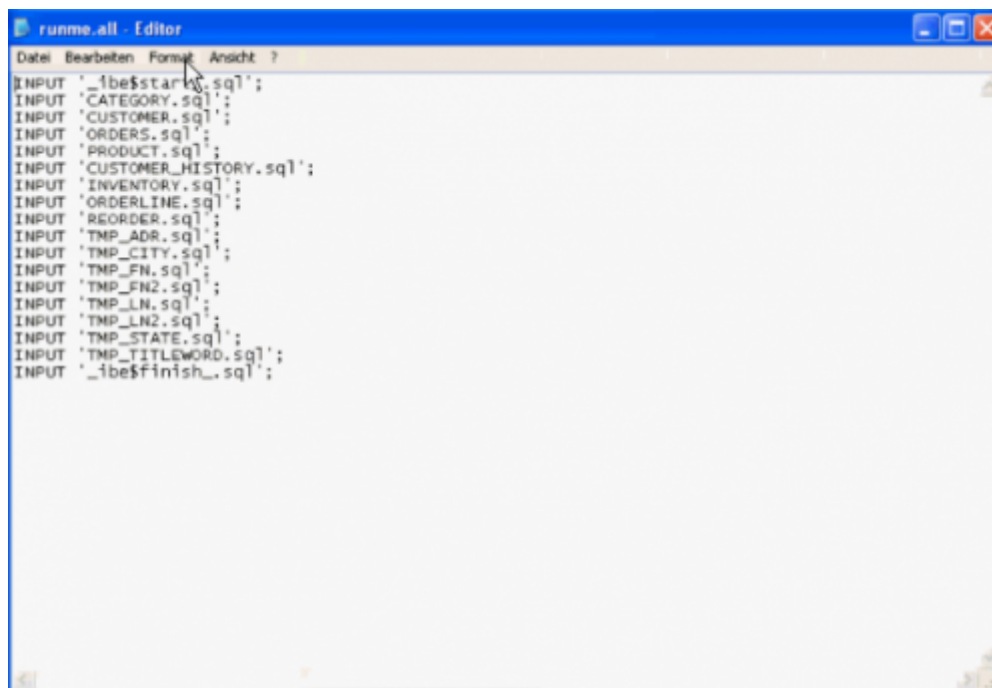
If you are already acquainted with IBExpert's *Extract Metadata* feature you will notice some keywords which are only supported by IBExpert e.g. `REINSERT`. When you are continually using the same `INSERT` statement, for example, `INSERT INTO CUSTOMER NAME1, NAME2` etc. the `INSERT INTO TABLE_NAME, COLUMN_NAME` part constantly has to be repeated; IBExpert directly replaces this using

REINSERT. So what you see when running the extract script is that there is only one full compatible insert statement, the rest is started by the keyword REINSERT and the list of values.

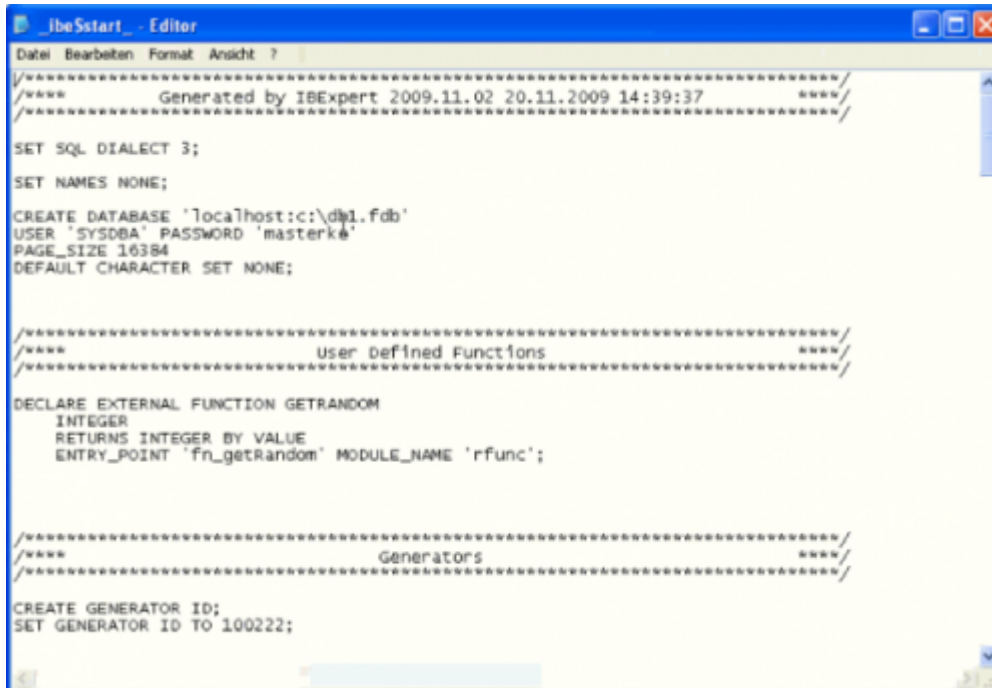
The speed of the extract depends on your machine. It can be 200-500% of a typical GBAK operation. But you have one big advantage, which becomes apparent when you take a look at the folder where the data has been extracted to:



namely, you have a collection of SQL scripts which are run in sequence by the runme.all file to restore the full metadata and data as specified. Any alterations you may wish to make can be done manually by replacing any of the database definitions in the files in this folder. The database can be recreated by the runme batch file:



IBExpert begins by executing `ibe$start.sql`:



```

_ibeStart_ - Editor
Datei Bearbeiten Format Ansicht ?

/*****
Generated by IBExpert 2009.11.02 20.11.2009 14:39:37
*****/

SET SQL DIALECT 3;
SET NAMES NONE;

CREATE DATABASE 'localhost:c:\dml.fdb'
USER 'SYSDBA' PASSWORD 'masterkey'
PAGE_SIZE 16384
DEFAULT CHARACTER SET NONE;

/*****
User Defined Functions
*****/

DECLARE EXTERNAL FUNCTION GETRANDOM
INTEGER
RETURNS INTEGER BY VALUE
ENTRY_POINT 'fn_getrandom' MODULE_NAME 'rFunc';

/*****
Generators
*****/

CREATE GENERATOR ID;
SET GENERATOR ID TO 100222;

```

Here we can manually change the character set from `NONE` to any other character set. We can replace the database owner and recreate the database with a new owner. The create procedure statements – we can see with the `INITALL` procedure – are really just small prototypes:

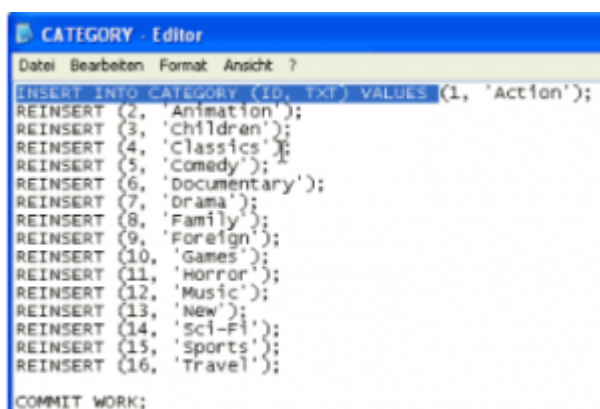
```

CREATE PROCEDURE INITALL(
    CNT BIGINT)
AS
BEGIN
EXIT;
END^

```

It already has the correct parameters but it has no body, because when you need to create a procedure which creates another procedure, one of the procedures has to be created first. As this is so often the case, one procedure calling another procedure which calls another procedure which calls the first one again, error messages are prevented by filling all these procedures with the body code later. We then define all tables, but without any primary or foreign key relations.

After all the objects have been created, the data is entered



```

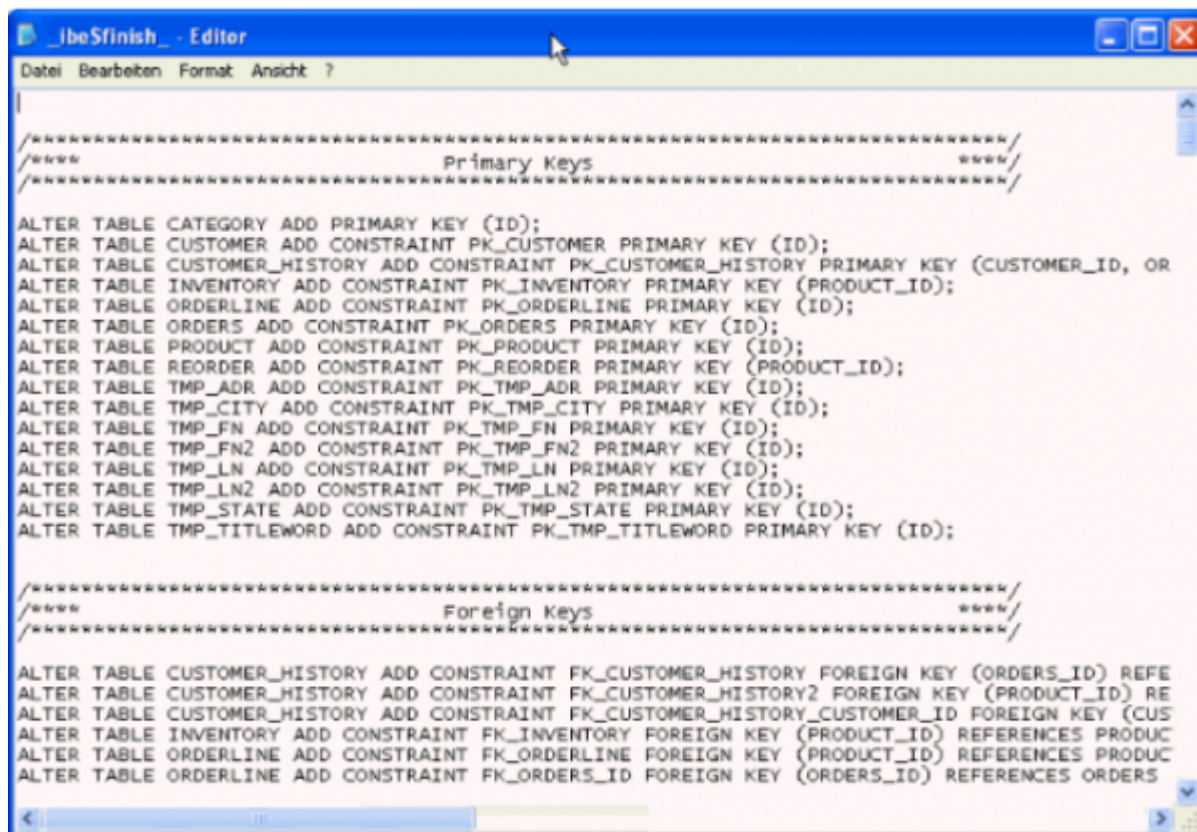
CATEGORY - Editor
Datei Bearbeiten Format Ansicht ?

INSERT INTO CATEGORY (ID, TXT) VALUES (1, 'Action');
REINSERT (2, 'Animation');
REINSERT (3, 'Children');
REINSERT (4, 'Classics');
REINSERT (5, 'Comedy');
REINSERT (6, 'Documentary');
REINSERT (7, 'Drama');
REINSERT (8, 'Family');
REINSERT (9, 'Foreign');
REINSERT (10, 'Games');
REINSERT (11, 'Horror');
REINSERT (12, 'Music');
REINSERT (13, 'New');
REINSERT (14, 'Sci-Fi');
REINSERT (15, 'Sports');
REINSERT (16, 'Travel');

COMMIT WORK;

```

from the script using `INSERT INTO` and `REINSERT`, which executes the last insert statement again. This is done for all tables, and in the last step, `ibe$finish`:



```

/IbeSfinish_ - Editor
Datei Bearbeiten Format Ansicht ?

/*****
Primary Keys
*****/

ALTER TABLE CATEGORY ADD PRIMARY KEY (ID);
ALTER TABLE CUSTOMER ADD CONSTRAINT PK_CUSTOMER PRIMARY KEY (ID);
ALTER TABLE CUSTOMER_HISTORY ADD CONSTRAINT PK_CUSTOMER_HISTORY PRIMARY KEY (CUSTOMER_ID, OR
ALTER TABLE INVENTORY ADD CONSTRAINT PK_INVENTORY PRIMARY KEY (PRODUCT_ID);
ALTER TABLE ORDERLINE ADD CONSTRAINT PK_ORDERLINE PRIMARY KEY (ID);
ALTER TABLE ORDERS ADD CONSTRAINT PK_ORDERS PRIMARY KEY (ID);
ALTER TABLE PRODUCT ADD CONSTRAINT PK_PRODUCT PRIMARY KEY (ID);
ALTER TABLE REORDER ADD CONSTRAINT PK_REORDER PRIMARY KEY (PRODUCT_ID);
ALTER TABLE TMP_ADR ADD CONSTRAINT PK_TMP_ADR PRIMARY KEY (ID);
ALTER TABLE TMP_CITY ADD CONSTRAINT PK_TMP_CITY PRIMARY KEY (ID);
ALTER TABLE TMP_FN ADD CONSTRAINT PK_TMP_FN PRIMARY KEY (ID);
ALTER TABLE TMP_FN2 ADD CONSTRAINT PK_TMP_FN2 PRIMARY KEY (ID);
ALTER TABLE TMP_LN ADD CONSTRAINT PK_TMP_LN PRIMARY KEY (ID);
ALTER TABLE TMP_LN2 ADD CONSTRAINT PK_TMP_LN2 PRIMARY KEY (ID);
ALTER TABLE TMP_STATE ADD CONSTRAINT PK_TMP_STATE PRIMARY KEY (ID);
ALTER TABLE TMP_TITLEWORD ADD CONSTRAINT PK_TMP_TITLEWORD PRIMARY KEY (ID);

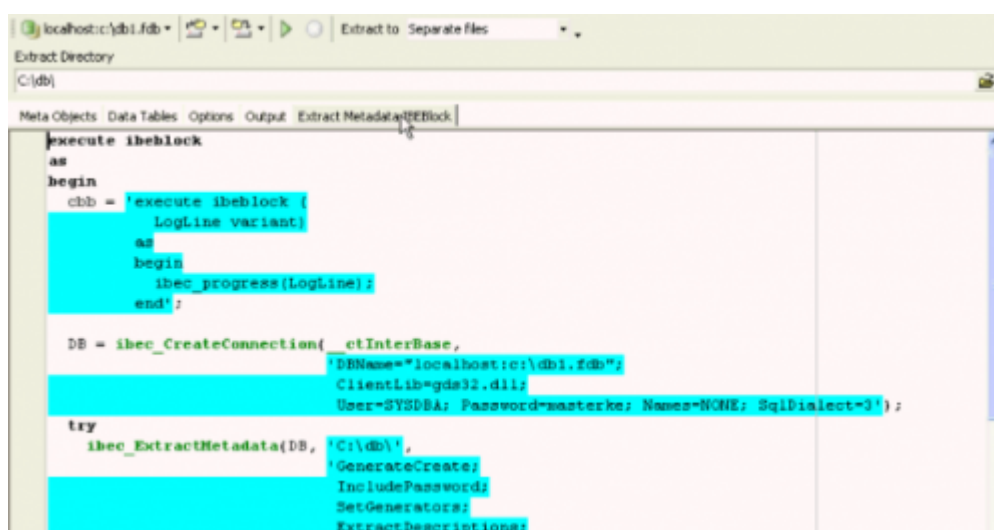
/*****
Foreign Keys
*****/

ALTER TABLE CUSTOMER_HISTORY ADD CONSTRAINT FK_CUSTOMER_HISTORY FOREIGN KEY (ORDERS_ID) REFE
ALTER TABLE CUSTOMER_HISTORY ADD CONSTRAINT FK_CUSTOMER_HISTORY2 FOREIGN KEY (PRODUCT_ID) RE
ALTER TABLE CUSTOMER_HISTORY ADD CONSTRAINT FK_CUSTOMER_HISTORY_CUSTOMER_ID FOREIGN KEY (CUS
ALTER TABLE INVENTORY ADD CONSTRAINT FK_INVENTORY FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUC
ALTER TABLE ORDERLINE ADD CONSTRAINT FK_ORDERLINE FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUC
ALTER TABLE ORDERLINE ADD CONSTRAINT FK_ORDERS_ID FOREIGN KEY (ORDERS_ID) REFERENCES ORDERS
  
```

all primary keys, foreign keys and indices are created, triggers and procedures are completed and privileges are restored.

Basically, anything happening in this series of scripts which prevents a restore, can be altered manually.

And if you have too little experience to recreate your database manually (it can be quite a dangerous undertaking!), you can use the [ExtractMetadataIBEBlock](#) feature, IBExpert's internal scripting language.



```

execute ibeblock
as
begin
  cbb = 'execute ibeblock {
    LogLine variant;
  }
  as
  begin
    ibec_progress(LogLine);
  end;

  DB = ibec_CreateConnection( ctInterBase,
    'DBName="localhost:c:\db1.fdb";
    ClientLib=gds32.dll;
    User=SYSDBA; Password=masterkey; Names=NONE; SqlDialect=3');

  try
    ibec_ExtractMetadata(DB, 'C:\db\1',
      'GenerateCreate;
      IncludePassword;
      SetGenerators;
      ExtractDescriptions;
  
```

Almost all the functionalities that are available in IBExpert can be directly executed from a script. When this script is copied into the SQL Editor and executed, it extracts the metadata and data and puts them into the specified directory. And you can execute this script not only from here, but also using IBExpert's command-line tool, [IBEScript](#):


```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Dokumente und Einstellungen\IBEXPERT>cd \
C:\>cd Programme\HK-Software\IBExpert
C:\Programme\HK-Software\IBExpert>ibscript
IBEScript Version 2009.11.02 Copyright (c) 2002-2009 IBExpert KG (www.ibexpert.com)

Syntax: IBEScript script_filename [options]

-S = silent mode
-U<verbose_file> = verbose output file.
  If <verbose_file> exists IBEScript will overwrite it
-v<verbose_file> = verbose output file.
  If <verbose_file> exists IBEScript will append messages to this file
-E = display only error messages
-M = continue after error
-T = write timestamp into log
-D = connection string (use it if your script doesn't contain CONNECT
  or CREATE DATABASE statements)
-P = connection password (use only with -D option)
-U = connection user name (use only with -D option)
-R = connection role (use only with -D option)
-C = charset (use only with -D option)
-l = client library file (gds32.dll if not specified)
-L{1|2|3} = SQL Dialect (use only with -D option; 1 if not specified)
-i = idle priority
-G<variable>=<value> = sets value of global variable (IBEBLOCK)

-----
-e = encrypt a script file (no execution will be performed)
-d = decrypt an encrypted script file (no execution will be performed)
-p<password> = encrypt/decrypt password
-o<file_name> = output file for encrypted and decrypted scripts

WARNING: All options are case-sensitive!

Example 1: IBEScript "C:\My Scripts\CreateDB.sql"
Example 2: IBEScript C:\MyScripts\CreateDB.sql -S -UScriptLog.txt

```

This functionality can also be helpful if you need to make some global replacements in the database as the Firebird server is, quite rightly, very strict about quite a lot of things. For example, if you want to rename a table column that is used a lot, it is often difficult to find and deactivate all references in the database to this column. This is one possible solution. When the references are only in certain stored procedures and in other databases, IBExpert offers the option to deactivate all, or a selection of stored procedures from the DB Explorer [right-click menu](#) - which avoids the error message, *object in use* when making metadata changes. Deactivate triggers and deactivate indices is already known. They can be reactivated in the same way.

The [ExtractMetadataIBEBLOCK](#) can also be used automatically in batch usage. For example, if you want to create a database from your development database, but without your test data, you simply extract all metadata, and add where clauses for your tables individually and recreate this database automatically from the script.

It is also a really easy way to downgrade your database. For example, you're working with an InterBase database and you want to replace this with Firebird. You can execute this extract metadata, take your script that defines all the data, and alter those things that are not compatible, for example InterBase allows you to use object names up to 64 chars, Firebird only uses 31 or 32 chars. So when you have long object names, you can simply replace the object names in the script files, and execute this script again and you can directly convert your InterBase database to a Firebird database.

A quick method to create a copy of your database, if you need 100% reproduction of the existing database is to use `CREATE SHADOW`:

```
CREATE SHADOW 1 'C:\DB1.FDX'
```

When this is committed, it immediately creates a physical copy of the database file. It is the fastest way to do a very dirty backup, it is however a solution if you have a time-frame problem.

Source: Firebird Conference 2009, session 14A Holger Klemt.

From:
<http://ibexpert.com/docu/> - **IBExpert**

Permanent link:
<http://ibexpert.com/docu/doku.php?id=02-ibexpert:02-08-ibexpert-tools-menu:extract-metadata:using-extract-metadata-to-repair-database>

Last update: **2023/10/06 18:20**

