SQLdb Tutorial0

From Free Pascal wiki

| **English (en)** | **français (fr)** | **日本語 (ja)** |

Contents

Database portal

References:

- General info
- Libraries
- Field types
- Controls
- FAQ
- SQL how-to
- Working With TSQLQuery
- In-memory database applications

Tutorials/practical articles:

- Overview
- 0 - Database set-up
- 1 - Getting started
- 2 - Editing
- 3 - Queries
- 4 - Data modules
- SQLdb Programming Reference

Databases

Advantage - MySQL - MSSQL - Postgres - Interbase - Firebird - Oracle - ODBC - Paradox - SQLite - dBASE - MS Access - Zeos

Introduction

This article will help you set up your database environment with sample tables and data to use in SQLdb Tutorial 1, 2 and 3.

Although this page is quite long, you will only go through a few of the sections as this article contains instructions for various database systems.

If you want to get up and running quickly, I'd suggest installing the Firebird server and example database as described below.

Requirements

This tutorial is written for use with recent Lazarus versions (Laz 1.0); it should also work on older versions Lazarus 0.9.30.

Furthermore you need an SQL/relational database, such as Firebird (http://sourceforge.net/project/showfiles.php?group_id=9028) (if possible version 2.0 or newer). It's easiest if you use standard settings (e.g user name SYSDBA and password masterkey), and that you have the employee sample database installed.

You can use another database (e.g. Microsoft SQL Server, MySQL, PostgreSQL, Oracle, SQLite, Sybase ASE or another database using ODBC). Please make sure you have the required database client libraries installed (see the various wiki articles on databases) Please see below (section **No Firebird or employee.fdb installed?**)

for a way to set up your tables

Firebird installation

In case you haven't yet installed Firebird or the sample database that is used with the tutorials, please find some instructions below.

Install Firebird client/server

If you haven't installed anything, you can download and run the installer from www.firebirdsql.org and install the server (e.g. "32-bit Classic, Superclassic & Superserver").

Firebird database libraries on Windows

After installation, on Windows you will need to have the Firebird client DLLs present:

- they could be in your system directory (available for all programs)
- they can also be in your Lazarus directory (for design time support in the IDE) and in the output directory where the executable is (to run the compiled program).

If you haven't installed the Firebird server, an easy way to get the client DLLs is: download Firebird Embedded 2.5 from [1] (http://www.firebirdsql.org/en/firebird-2-5/) Extract these files to your application directory:

```
fbclient.dll #only if using the server
#or
fbembed.dll #only if using embedded
firebird.msg
ib_util.dll
icudt30.dll
icuin30.dll
icuuc30.dll
Microsoft.VC80.CRT.manifest
msvcp80.dll
msvcr80.dll
```

Rename fbembed.dll to fbclient.dll (the name for a regular, client-server Firebird client - this helps use on older Lazarus/FPC versions). The embedded Firebird DLL can also act as a regular Firebird client.

Make sure the employee.fdb database is in your project directory. You can copy it from the examples/empbuild/ directory of the firebird 2.5 server.

Finally, compile your project (even if it empty) once to create the output directory, and copy the dlls, as well as the employee.fdb database, into that directory.

Firebird database libraries on other systems

On Linux/OSX, you will also need the Firebird client shared libraries. On Linux you can use your distribution's method of getting programs to get the Firebird client packages, e.g. on Debian:

```
aptitude install libfbclient2 firebird-dev #we need the dev version because FPC 2.6 and lower will look for
libfbclient.so
```

No Firebird or employee.fdb installed?

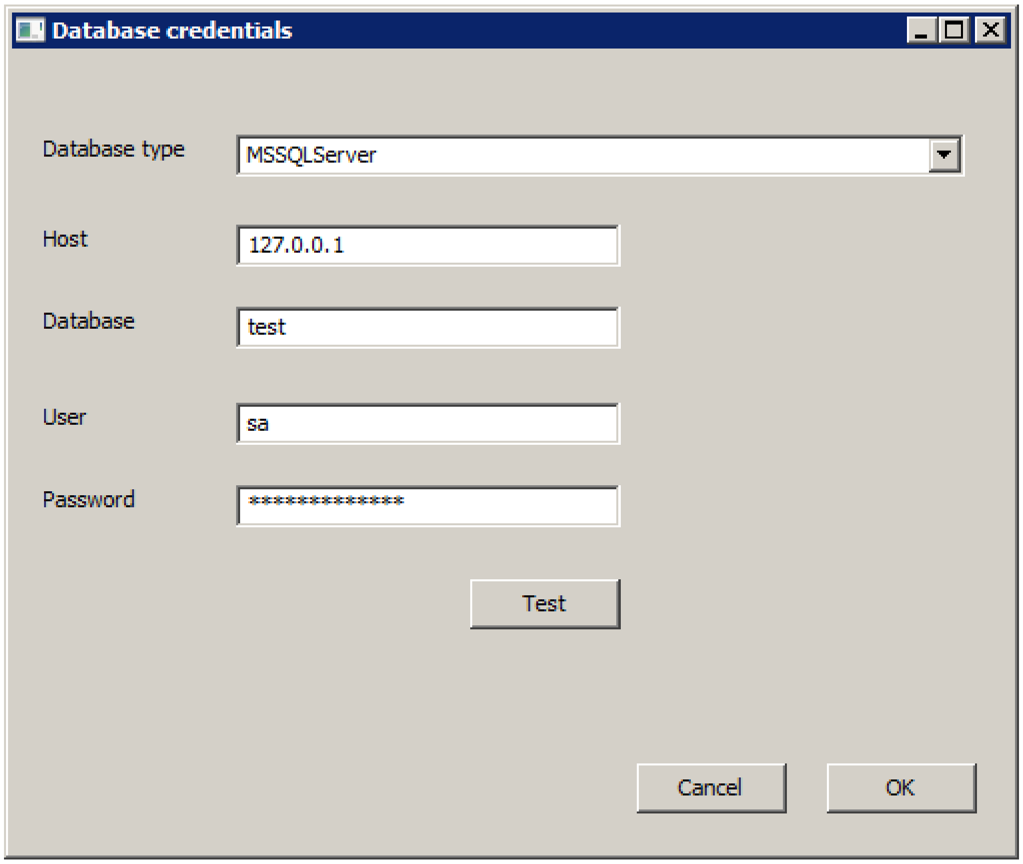
If you don't have the employee sample database installed or are using a different database, here is a minimal version of the table we'll be using (note: directions for some specific databases can be found below).

Automatic creation

The easiest way of setting this up is to create a new empty database on your system, then connect to it with the TSQLScriptSample sample/utility program. This program is included with current Lazarus development releases in the `./examples/database/tsqlscript` directory, but can also be downloaded from

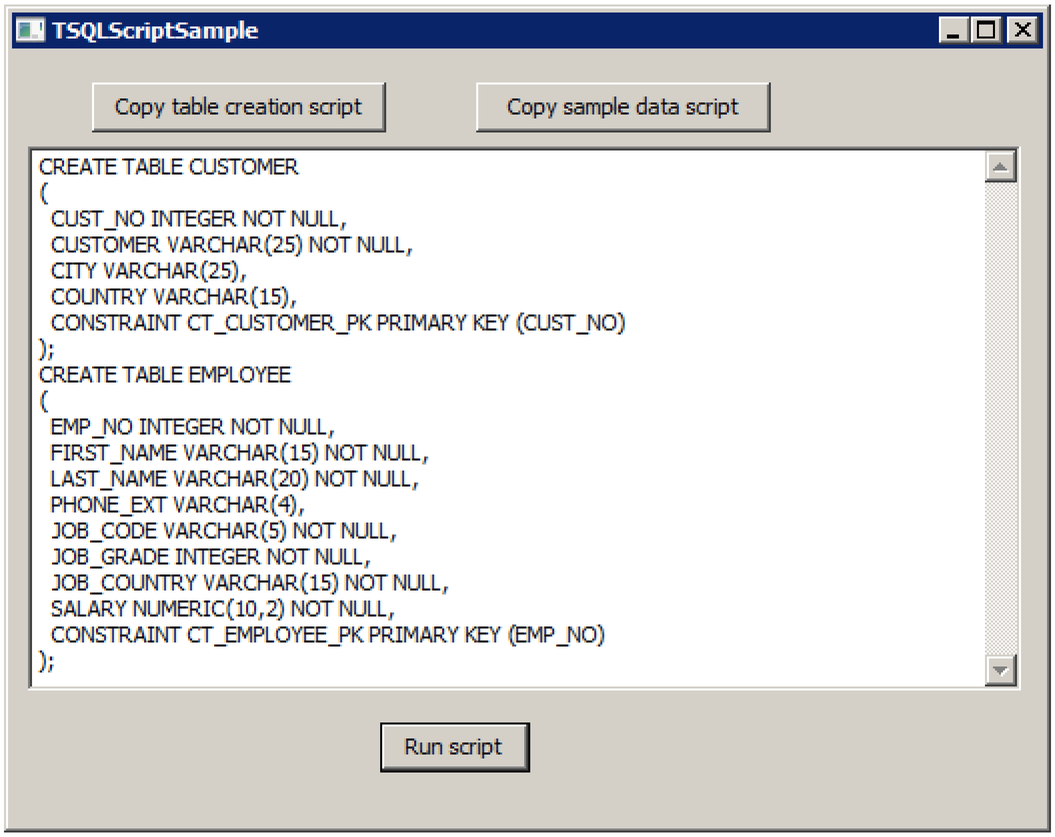
> http://svn.freepascal.org/svn/lazarus/trunk/examples/database/tsqlscript/

Compile and run the program, then connect to the database:



Database type: MSSQLServer
Host: 127.0.0.1
Database: test
User: sa
Password: **************

then click the button *Copy table creation script*, and *Run script*:



```
CREATE TABLE CUSTOMER
(
 CUST_NO INTEGER NOT NULL,
 CUSTOMER VARCHAR(25) NOT NULL,
 CITY VARCHAR(25),
 COUNTRY VARCHAR(15),
 CONSTRAINT CT_CUSTOMER_PK PRIMARY KEY (CUST_NO)
);
CREATE TABLE EMPLOYEE
(
 EMP_NO INTEGER NOT NULL,
 FIRST_NAME VARCHAR(15) NOT NULL,
 LAST_NAME VARCHAR(20) NOT NULL,
 PHONE_EXT VARCHAR(4),
 JOB_CODE VARCHAR(5) NOT NULL,
 JOB_GRADE INTEGER NOT NULL,
 JOB_COUNTRY VARCHAR(15) NOT NULL,
 SALARY NUMERIC(10,2) NOT NULL,
 CONSTRAINT CT_EMPLOYEE_PK PRIMARY KEY (EMP_NO)
);
```

The program should indicate success. Now do the same for *Copy sample data script*, and *Run script*

In case of problems, you can try the manual steps below.

Manual creation

We will create CUSTOMER and EMPLOYEE tables that will be used in a later tutorial.

Run these SQL commands in your database editor/tool:

```
CREATE TABLE CUSTOMER
(
  CUST_NO INTEGER NOT NULL,
  CUSTOMER VARCHAR(25) NOT NULL,
  CITY VARCHAR(25),
  COUNTRY VARCHAR(15),
  CONSTRAINT CT_CUSTOMER_PK PRIMARY KEY (CUST_NO)
);
```

```
CREATE TABLE EMPLOYEE
(
  EMP_NO INTEGER NOT NULL,
  FIRST_NAME VARCHAR(15) NOT NULL,
  LAST_NAME VARCHAR(20) NOT NULL,
  PHONE_EXT VARCHAR(4),
  JOB_CODE VARCHAR(5) NOT NULL,
  JOB_GRADE INTEGER NOT NULL,
  JOB_COUNTRY VARCHAR(15) NOT NULL,
  SALARY NUMERIC(10,2) NOT NULL,
  CONSTRAINT CT_EMPLOYEE_PK PRIMARY KEY (EMP_NO)
);
```

Some data so you can at least show something.... first some clients:

```
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (1, 'Michael Design', 'San Diego', 'USA');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (2, 'VC Technologies', 'Dallas', 'USA');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (3, 'Klämpfl, Van Canneyt', 'Boston', 'USA');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (4, 'Felipe Bank', 'Manchester', 'England');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (5, 'Joost Systems, LTD.', 'Central Hong Kong',
'Hong Kong');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (6, 'Van der Voort Int.', 'Ottawa', 'Canada');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (7, 'Mrs. Mauvais', 'Pebble Beach', 'USA');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (8, 'Asinine Vacation Rentals', 'Lihue', 'USA');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (9, 'Fax', 'Turtle Island', 'Fiji');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (10, 'FPC Corporation', 'Tokyo', 'Japan');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (11, 'Dynamic Intelligence Corp', 'Zurich',
'Switzerland');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (12, '3D-Pad Corp.', 'Paris', 'France');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (13, 'Swen Export, Ltd.', 'Milan', 'Italy');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (14, 'Graeme Consulting', 'Brussels', 'Belgium');
INSERT INTO CUSTOMER (CUST_NO, CUSTOMER, CITY, COUNTRY) VALUES (15, 'Klenin Inc.', 'Den Haag', 'Netherlands');
```

Then some employees:

```
INSERT INTO EMPLOYEE (emp_no, first_name, last_name, phone_ext, job_code, job_grade,
  job_country, salary)
  VALUES (1,'William','Shatner','1702','CEO',1,'USA',48000);
INSERT INTO EMPLOYEE (emp_no, first_name, last_name, phone_ext, job_code, job_grade,
  job_country, salary)
  VALUES (2,'Ivan','Rzeszow','9802','Eng',2,'Russia',38000);
INSERT INTO EMPLOYEE (emp_no, first_name, last_name, phone_ext, job_code, job_grade,
  job_country, salary)
  VALUES (3,'Erin','Powell','1703','Admin',2,'USA',45368);
```

Please create the database, table and insert the data in your database environment.

SQLite

If you are using SQLite, you can create the database mentioned above in your project directory by running the sqlite executable:

```
sqlite employee.sqlite
```

Now copy and paste the above CREATE TABLE and INSERT statements. To test if the right data is present, enter this query:

```
select * from customer;
```

End your session with

```
.quit
```

A file called employee.sqlite should now be created in your project directory.

Make sure the required sqlite dll/so is installed - e.g. on Windows, sqlite3.dll should be present in

- your Lazarus + project output directory or
- your system directory

Also, if you have a 32-bit sqlite dll, make sure you have a 32-bit lazarus.

Compile your project (even if it is empty) once to create the output directory, and (on Windows) copy the dll, as well as the employee.sqlite database, into that directory.

PostgreSQL

This section assumes you're using a Linux server and the shell; comparable steps can be done using Windows and GUI tools such as pgadmin.

Log in to your server and switch to the postgres account:

```
su - postgres -c psql # immediately start up psql SQL interpreter
```

Create a user for the database and the tables:

```
CREATE USER employee WITH PASSWORD 'hellopassword'; -- of course, adjust password to taste
-- something like  'CREATE ROLE' should appear indicating success.
-- to later change the password you can use something like
-- alter user employee with password '<newpasswordhere>';

-- We're going to let the password never expire; if you want more security, you can leave this step out:
ALTER USER employee VALID UNTIL 'infinity'; --password never expires

-- Now we're tightening it up a bit again:
-- Don't allow user to create a database or create other users:
ALTER USER employee NOCREATEDB NOCREATEROLE; --restrict object creation
-- something like 'ALTER ROLE' should appear indicating success.

-- Create our database:
CREATE DATABASE employee;
-- something like CREATE DATABASE should appear indicating success.

-- Assign all privileges on database employee to user employee:
GRANT ALL PRIVILEGES ON DATABASE employee TO employee; -- allow user full permissions to database
-- something like GRANT should appear indicating success.

-- We create the table using a serial datatype - aka autonumber/autoincrement:
CREATE TABLE customer
(
  cust_no serial NOT NULL,
  customer character varying(25) NOT NULL,
  city character varying(25),
  country character varying(15),
  CONSTRAINT integ_60 PRIMARY KEY (cust_no )
);

-- Then create the employee table:
CREATE TABLE EMPLOYEE
(
  EMP_NO SERIAL NOT NULL,
  FIRST_NAME VARCHAR(15) NOT NULL,
  LAST_NAME VARCHAR(20) NOT NULL,
  PHONE_EXT VARCHAR(4),
  JOB_CODE VARCHAR(5) NOT NULL,
  JOB_GRADE INTEGER NOT NULL,
  JOB_COUNTRY VARCHAR(15) NOT NULL,
  SALARY NUMERIC(10,2) NOT NULL,
  CONSTRAINT CT_EMPLOYEE_PK PRIMARY KEY (EMP_NO)
);

-- Now copy and paste the INSERT statements from the section above to insert the data.

-- To test if the right data is present, enter this query:
SELECT * FROM customer;
-- You should see some customer data.

-- Exit out of psql:
```

```
\q
```

Now you should be on a shell logged in as the postgres user.

If your server is on another machine than your development machine, make sure you allow network access to the database. See your postgresql documentation for details, but something like this should work:

```
# please adjust nano (e.g. use vim,emacs,joe...) and the postgres version number depending on situation
nano /etc/postgresql/8.4/main/pg_hba.conf
```

Verify if there is a line like - NOTE: replace 192.168.0.1 with your own LAN ip address range

#allow access from local network using md5 hashed passwords: host all all 192.168.0.1/24 md5

or more restrictive:

# only allow network access to the employee database by the employee user host employee employee 192.168.0.1/24 md5

If there isn't such a line, add the line at the end, save and close your editor. See PostgreSQL documentation for more details.

Reload PostgreSQL settings:

```
psql
```

then

```
SELECT pg_reload_conf(); --reload settings...
-- ...and exit back to shell:
\q
```

Test logging in to PostgreSQL.

Note: by default PostgreSQL tries an ident/unix domain socket login which doesn't allow passwords. So we specify a host to force TCP/IP login:

```
psql -h 127.0.0.1 -d employee -U employee -W #Log in via tcp/ip. Enter your db password.
```

Make sure the required PostgreSQL dll/so and any required other libraries is installed - e.g. on Windows, they should be either:

- in your Lazarus + project output directory or
- in your system directory

Compile your project (even if it is empty) once to create the output directory, and (on Windows) copy the dlls into that directory.

Finish

Now you've set up your database, you can continue to the first "real" tutorial.

See also

- SQLdb Tutorial1: First part of the DB tutorial series, showing how to set up a grid that shows database data
- SQLdb Tutorial2: Second part of the DB tutorial series, showing editing, inserting etc.
- SQLdb Tutorial3: Third part of the DB tutorial series, showing how to program for multiple databases and use a login form
- SQLdb Tutorial4: Fourth part of the DB tutorial series, showing how to use data modules
- Lazarus Database Overview: Information about the databases that Lazarus supports. Links to database-specific notes.
- SQLdb Package: information about the SQLdb package
- SQLdb Programming Reference: an overview of the interaction of the SQLdb database components
- SqlDBHowto: information about using the SQLdb package
- Working With TSQLQuery: information about TSQLQuery
