Руководство по установке и эксплуатации

Release 9.0.0

pgCodeKeeper

Dec 07, 2023

Оглавление

The pgCodeKeeper plug-in is designed for comparing PostgreSQL DB schemas and selective applying the differences, considering possible dependences between the schema objects.

Work in pgCodeKeeper is conducted with the help of projects. A project is represented by DB schema objects, stored in separate files with hierarchical structure.

Using the project, it is possible to view the structure of files-objects and create a project scheme migration script for DB or vice versa, modify the project structure with objects from a DB.

pgCodeKeeper source code is available at GitHub for free.

Глава

ONE

PREFACE

When I started working in Technology LTD in 2013, the company had already been developing software for one of the most leading taxi booking services in Russia - Maxim. And one of the tasks of that time was to start using PostgreSQL database server instead of the MSSQL.

We started the migration process from one database management system to another and discovered two key features of PostgreSQL.

First, set of tools for both development and maintenance of databases at that time was vastly inferior to the well-known MSSQL.

Second, an architecture of the existing information system included a significant overlap of application business logic resided in the database. These are database views related to a notable number of tables and stored procedures contained there.

We faced with the significant complexity of the migration process because, unlike DBMS such as MSSQL or Oracle, which recompile invalidated database objects, PostgreSQL acted in a very different manner.

Our worst nightmare becomes a hell of dependencies between objects. While of recreating an object, if it has a reference from other database objects, we had to delete all the dependent objects and create them again after recreating the parent object.

It was then when we had an idea of creating our own product, which first of all would be able to solve the problem occurring while forming the migration script with dependent objects, and above that - able to ease the work of the developer in many other ways such as: code completion, code navigation, forming of scripts with test data for existing tables etc.

Currently, pgCodeKeeper solves most of the problems at hand. But we are not planning to stop here. Next, we are aimed at both improving the existing possibilities and achieving new goals.

We work hard for pgCodeKeeper to become a tool for database objects refactoring, as well as for improving work with the pl/pgSql code, and retrieving and displaying data.

Now that we have pgCodeKeeper, working with PostgreSQL has become significantly easier. In late 2017, it was decided to make our product code subject to one of the existing open source licenses. We went for the Apache 2.0. license. And thanks to that, pgCodeKeeper is now available for many participants all over the world.

I thank you for the interest you have shown in our product and wish you pleasant work with pgCodeKeeper.

Head of software development, Naberezhnye Chelny

Andrew G. Saushkin, 2018

Глава

TWO

INSTALLATION

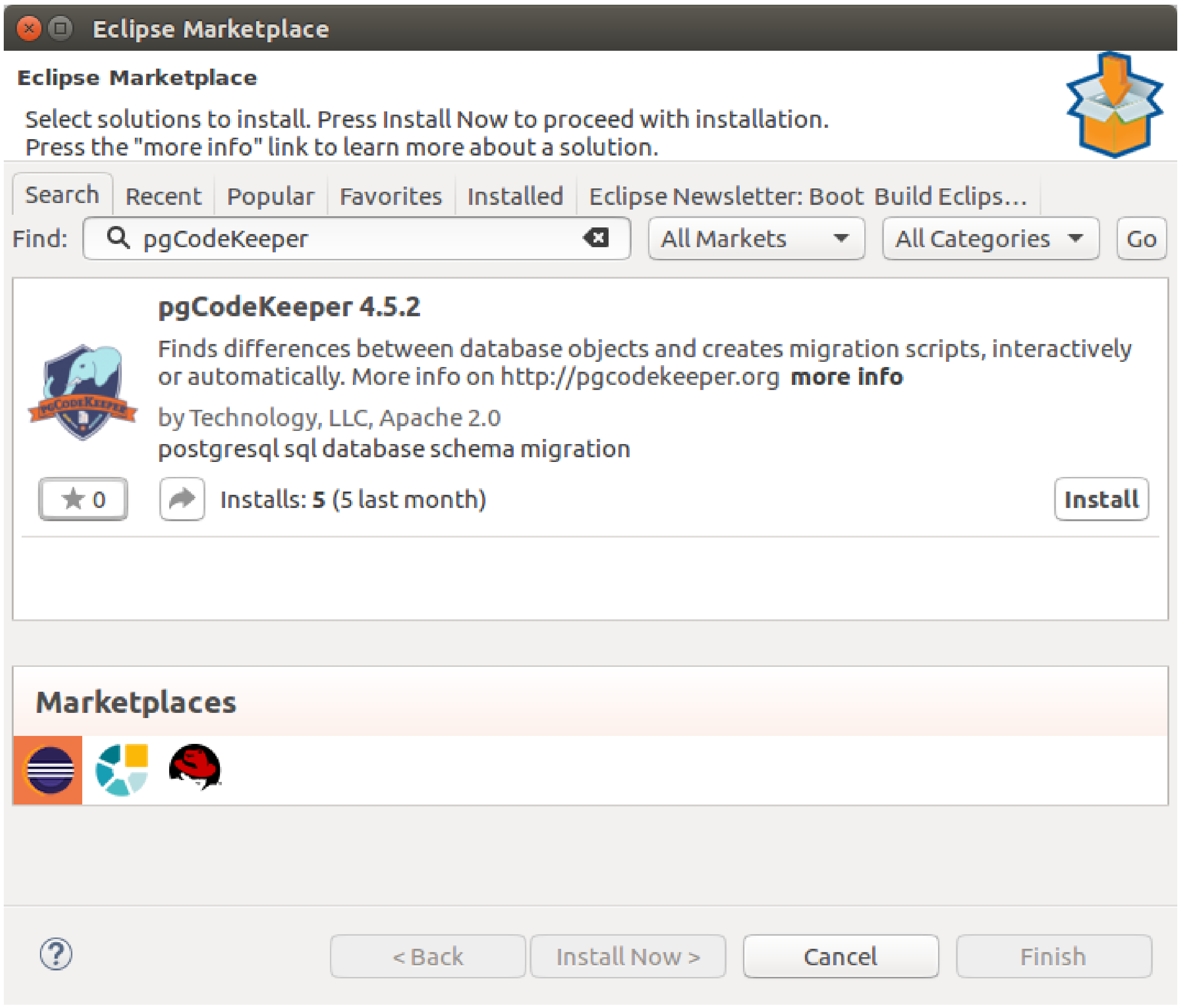
First, install Java SDK version 11+ for your platform. The latest Eclipse versions require the 17+ for proper functioning

2.1 Eclipse Marketplace

The easiest way to install pgCodeKeeper is Eclipse Marketplace: launch Eclipse, version 4.6 or higher, with the installed Eclipse Marketplace, and select Help > Eclipse Markeplace. . . .

Enter “pgCodeKeeper” into the search box.

Select the pgCodeKeeper package, review the Terms of service, and click Finish for the installation of the latest pgCodeKeeper version to begin.



Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

pgCodeKeeper 4.5.2

Finds differences between database objects and creates migration scripts, interactively or automatically. More info on http://pgcodekeeper.org **more info**

by Technology, LLC, Apache 2.0

postgresql sql database schema migration

Installs: **5** (5 last month)

After the installation is finished, restart Eclipse to apply the changes.

Launch Eclipse and select Window > Perspective > Open Perspective > Other > pgCodeKeeper.

To download a previous pgCodeKeeper version, go to https://github.com/pgcodekeeper/pgcodekeeper/releases, download the required release version, and install it manually. On the website, you can find release versions starting with 5.10.5.

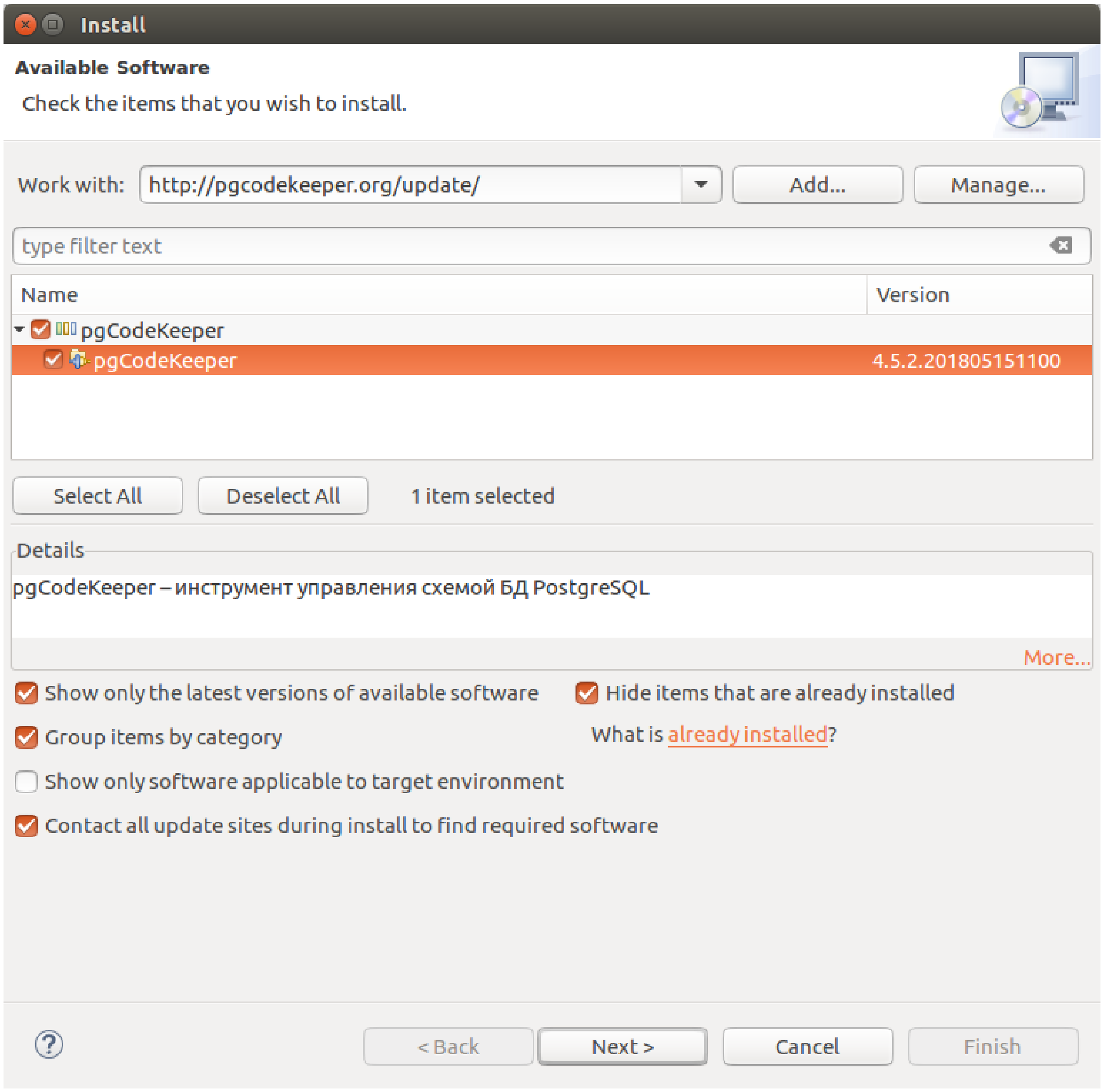
2.2 Update website

Another way to install pgCodeKeeper is the update website: launch Eclipse, version 4.6 or higher, select Help > Install New Software… Plug-in installation (with the help of the update web-site) wizard will appear on the screen.

In the Work With field specify the path to the update website: http://pgcodekeeper.org/update/.

After pressing Enter you'll see the available plug-ins in the list below.

Select the pgCodeKeeper package, agree to the installation, review the Terms of service and click Finish for the installation to begin.



Available Software

Check the items that you wish to install.

Work with: http://pgcodekeeper.org/update/

Name	Version
✓ pgCodeKeeper	
✓ pgCodeKeeper	4.5.2.201805151100

1 item selected

Details

pgCodeKeeper – инструмент управления схемой БД PostgreSQL

✓ Show only the latest versions of available software

✓ Hide items that are already installed

What is already installed?

✓ Group items by category

☐ Show only software applicable to target environment

✓ Contact all update sites during install to find required software

After the installation is finished, restart Eclipse to apply the changes.

Launch Eclipse and select Window > Perspective > Open Perspective > Other > pgCodeKeeper.

2.3 pgCodeKeeper standalone build

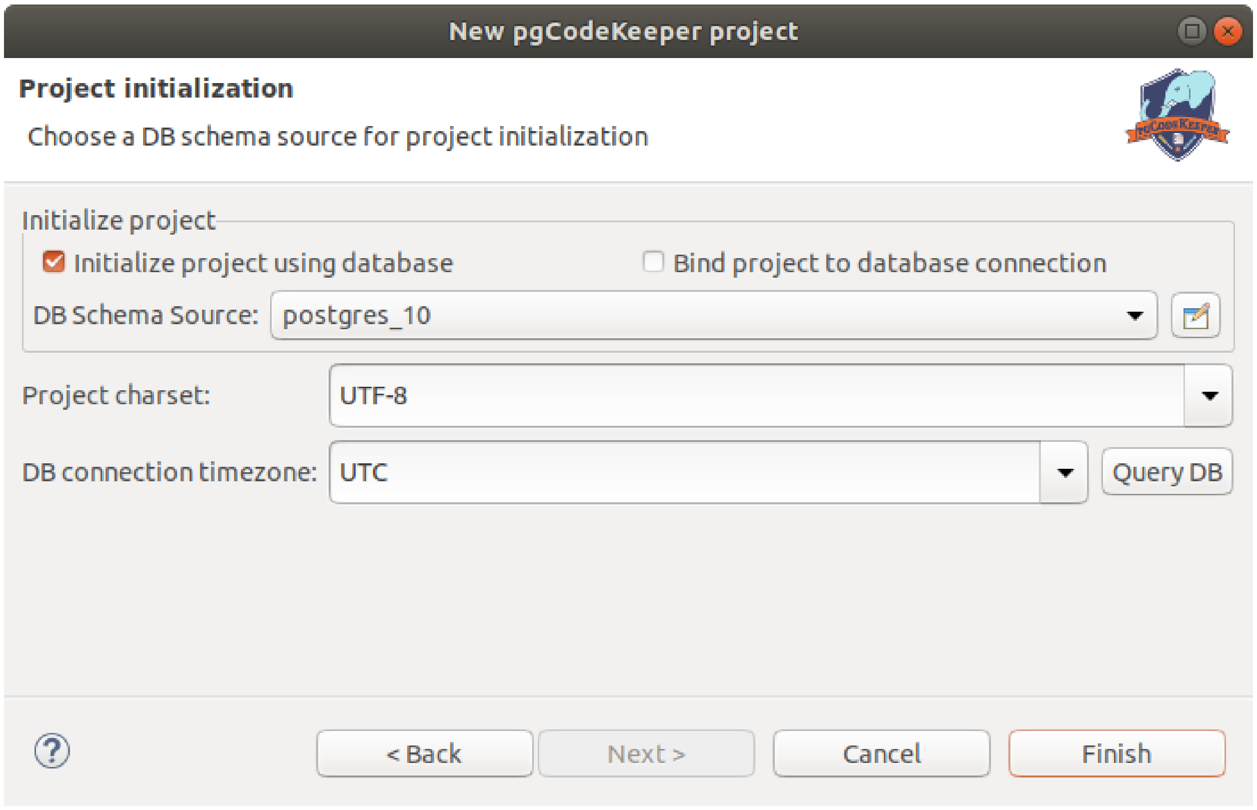
Another alternative is pgCodeKeeper standalone build that you can download here.

Глава

THREE

CREATING A NEW PROJECT

- Start the new project creation wizard: File-> New-> Project…
- In the pgCodeKeeper category, select pgCodeKeeper Project to work with PostgreSQL or pgCodeKeeper MS SQL Project to work with MS SQL. Click Next.
- Enter the project name.
- Change the location and workspace of the project, if necessary.
- Click Next.


New pgCodeKeeper project

Project initialization

Choose a DB schema source for project initialization

Initialize project

✓ Initialize project using database

☐ Bind project to database connection

DB Schema Source: postgres_10

Project charset: UTF-8

DB connection timezone: UTC

- Select the database source or disable the Initialize project using database parameter.
- Select Bind project to database connection if you want to work with only a single database.
- Select the project's encoding.
- Select the timezone of the DB connection (you can get it from the current DB). This step is not available for MS SQL projects.

- Click Finish.

After a short wait, a new project appears and pgCodeKeeper project editor opens automatically.

Глава

FOUR

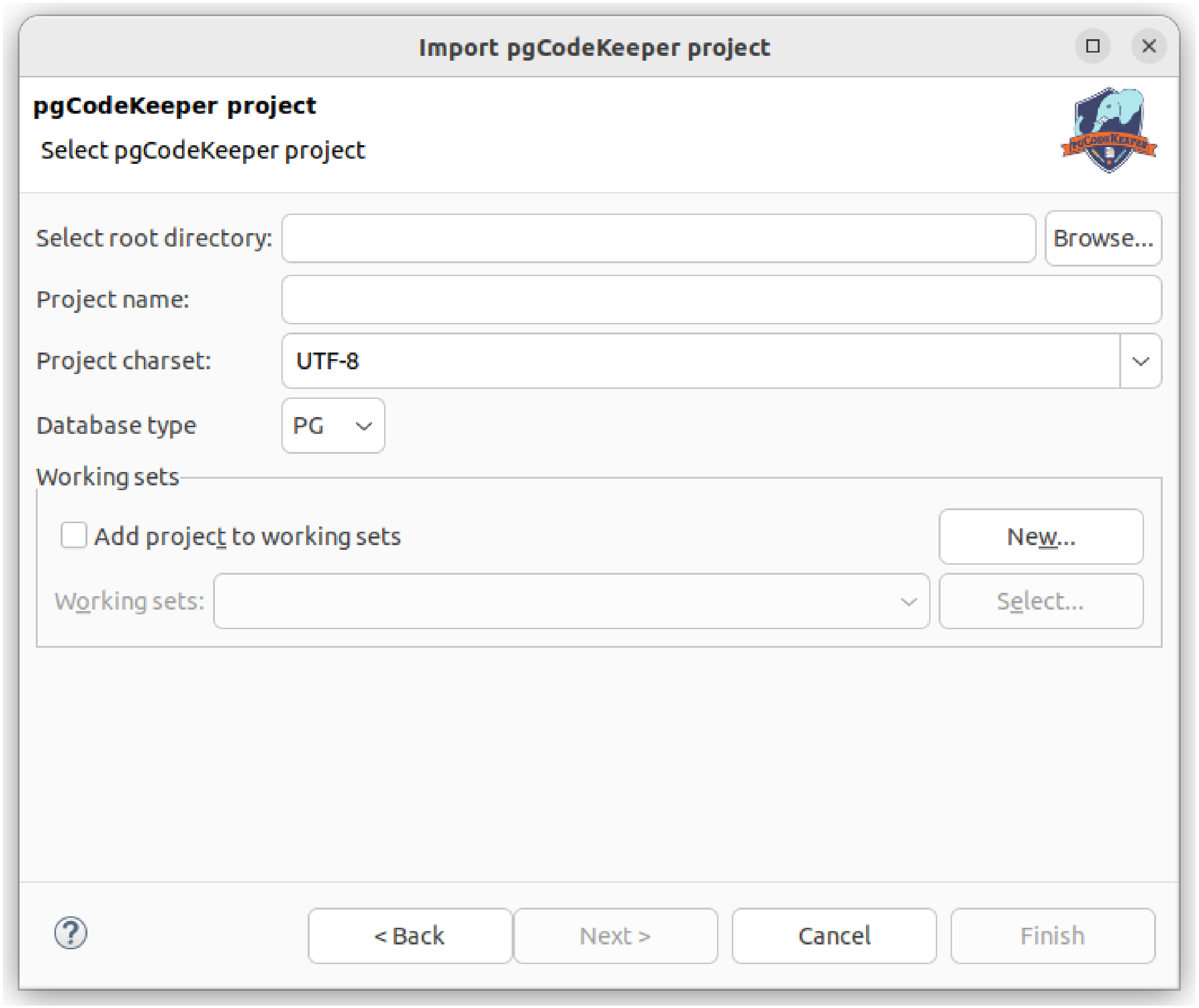
PROJECT IMPORT

If there is an Eclipse project (such projects have the .project file in the root), then import of the existing project is conducted:

1. Select File -> Import… from the menu. An import wizard selection screen will appear.
2. Select General -> Existing Project into Workplace and click Next.
3. Select the project directory.
4. Then, select the projects which you want to import.
5. Finish the import by clicking Finish.

If there is only a pgCodeKeeper project structure, without the Eclipse project files, importing is done with the help of the import wizard. Storing your pgCodeKeeper project structure without the Eclipse information is a common practice, for example, if you use a version control system.

1. Select File -> Import from the menu … An import wizard selection screen will appear.
2. Select pgCodeKeeper Project in the pgCodeKeeper category and click Next.



3. Specify path to the directory containing the pgCodeKeeper project structure, enter the project's name and select its encoding.
4. Select the Database type for the imported project.
5. To add the project to the workspace, select the corresponding option.
6. Click Finish.

For the correct work, the project should contain the '.pgcodekeeper' file in the root directory. If this file doesn't exist, project Import Wizard will automatically create it.

Глава

FIVE

PROJECT STRUCTURE

The PostgreSQL project has the following structure:

```
.
├─ EXTENSION
└─ SCHEMA
  ├─ schema_name
  ├─ ...
  └─ another_schema_name
    ├─ another_schema_name.sql
    ├─ DOMAIN
    ├─ FTS_CONFIGURATION
    ├─ FTS_DICTIONARY
    ├─ FTS_PARSER
    ├─ FTS_TEMPLATE
    ├─ FUNCTION
    ├─ OPERATOR
    ├─ PROCEDURE
    ├─ SEQUENCE
    ├─ TABLE
    │ └─ table_name.sql
    ├─ TYPE
    │ └─ type_name.sql
    └─ VIEW
      └─ view_name.sql
```

- schema_name . . . another_schema_name - schema names.
- another_schema_name.sql - an obligatory file with schema definition, the name should be identical to the directory name.
- table_name.sql, type_name.sql, view_name.sql - files with objects' definition. The names are specified without schema names.
- all the overloaded functions are stored in a single file. For example, the functions public.func(int) and public.func(text) will be located in SCHEMA/public/FUNCTION/func.sql.
- definition of indexes, constraints, triggers and rules are located in the files of parent objects. For example, index public.ii for the public.t1 table will be located in SCHEMA/public/TABLE/t1.sql.

The MS SQL project has the following structure:

```
.
├─ Assemblies
```

(continues on next page)

(continued from previous page)

```
├─ Functions
├─ Security
│  ├─ Roles
│  ├─ Schemas
│  │  └─ schema_name.sql
│  └─ Users
├─ Sequences
├─ Stored Procedures
├─ Tables
│  └─ schema_name.table_name.sql
├─ Types
│  └─ schema_name.type_name.sql
└─ Views
   └─ schema_name.view_name.sql
```

- schema_name.sql - an obligatory file with schema definition.
- schema_name.table_name.sql, schema_name.type_name.sql, schema_name.view_name.sql - files with object definitions объектов. Names are specified with the schema name.
- definitions of indexes, constraints, triggers are located in the files of parent objects. For example, index dbo.ii for the dbo.t1 table will be located in Tables/dbo.t1.sql.

Apart from this, a project may contain the following directories:

MIGRATION. Contains the saved migration scripts. You can select automatic save and deletion of migration scripts for this directory on the settings page DB update.

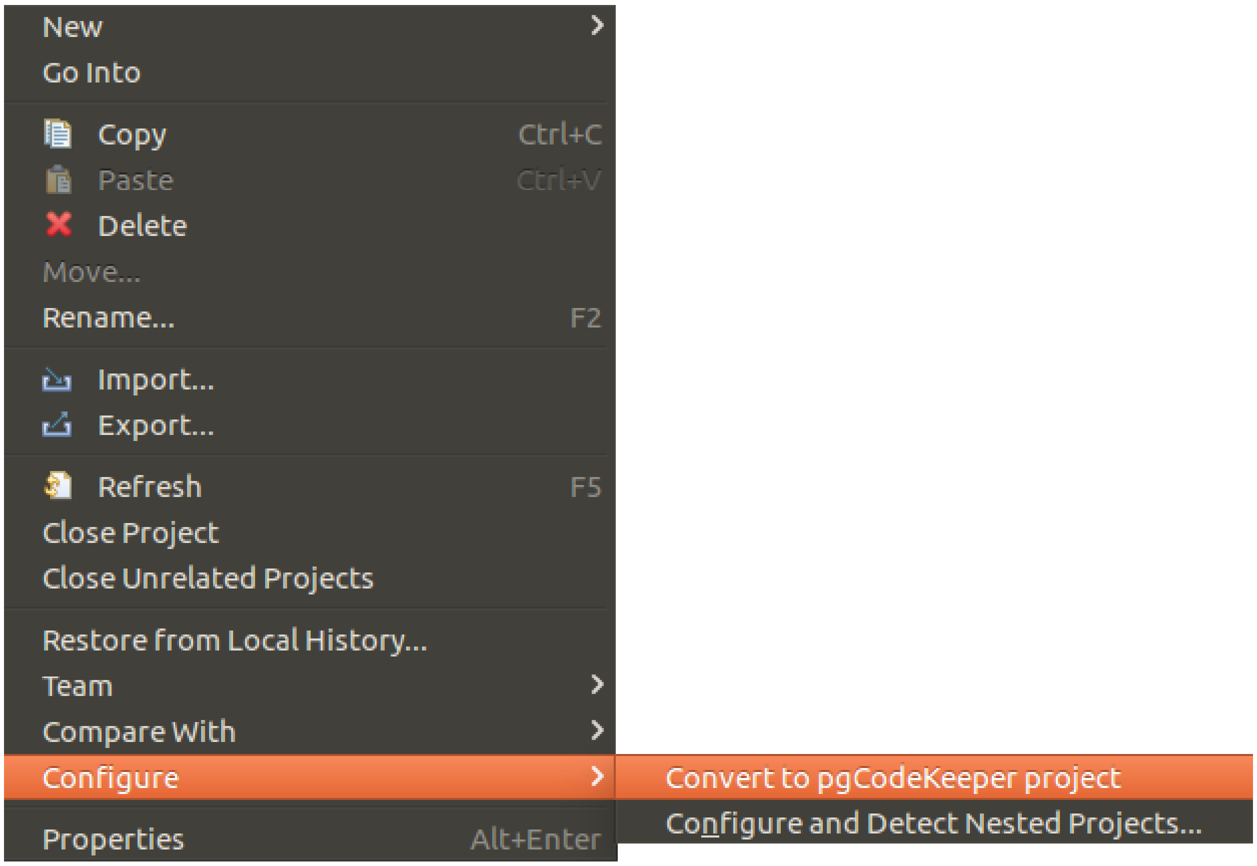
OVERRIDES. Contains files of object properties override. The inner structure of this directory duplicates the project structure. You can save the properties override by selecting the corresponding item while saving the objects in the project. Overriding privileges and owners of database objects is currently supported.

Глава

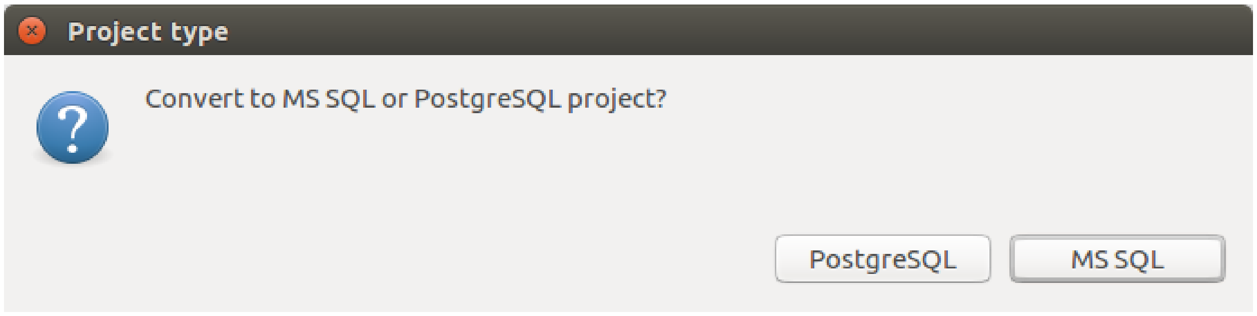
SIX

PROJECT CONVERTING

Every project can be converted into a pgCodeKeeper project. To do this, select Configure -> Convert to pgCodeKeeper project. in the project menu.

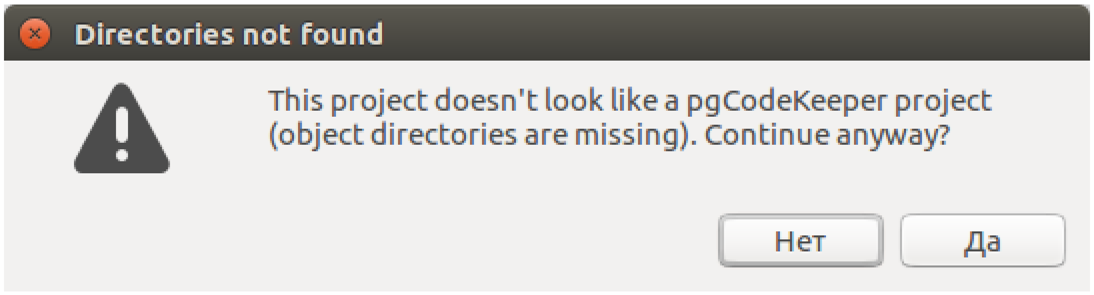


In some cases, you'll need to select the type of project: PostgreSQL or MS SQL.


Project type

Convert to MS SQL or PostgreSQL project?

If the structure is not up to standard, a notification will appear.


Directories not found

This project doesn't look like a pgCodeKeeper project (object directories are missing). Continue anyway?

Глава

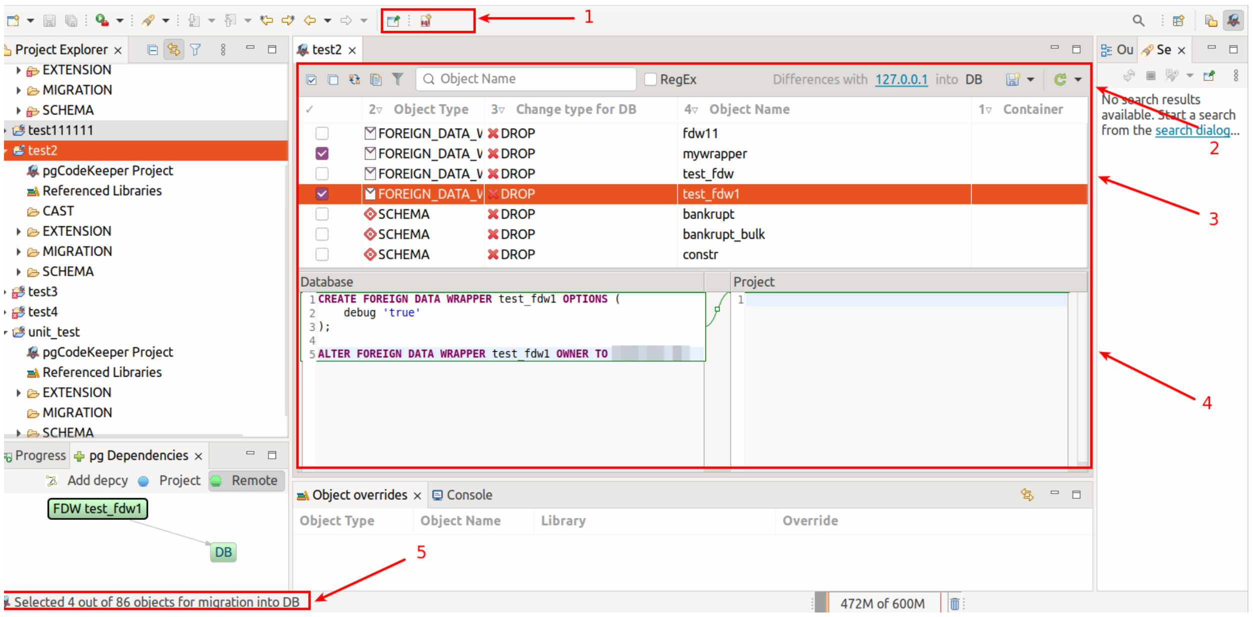
SEVEN

EDITORS AND VIEWS

7.1 pgCodeKeeper project editor

Project editor is the main way of working with pgCodeKeeper. The editor interface consists of several main parts:

1. Eclipse tool bar
2. Editor tool bar
3. Differences table
4. Comparison panel
5. Object counter



✓	2▿ Object Type	3▿ Change type for DB	4▿ Object Name	1▿ Container
☐	FOREIGN_DATA_V	DROP	fdw11	
✓	FOREIGN_DATA_V	DROP	mywrapper	
☐	FOREIGN_DATA_V	DROP	test_fdw	
✓	FOREIGN_DATA_V	DROP	test_fdw1	
☐	SCHEMA	DROP	bankrupt	
☐	SCHEMA	DROP	bankrupt_bulk	
☐	SCHEMA	DROP	constr	

```
CREATE FOREIGN DATA WRAPPER test_fdw1 OPTIONS (
    debug 'true'
);

ALTER FOREIGN DATA WRAPPER test_fdw1 OWNER TO [redacted]
```

Selected 4 out of 86 objects for migration into DB

Eclipse tool bar contains tools for working with SQL editor. The button allows you to open a new SQL editor.

Editor tool bar is used for working with the object list. There are the following opportunities:

- – select all objects.
- – deselect all the objects.

- – invert the selecting of objects.
- – copy the set of selected objects to the clipboard as a regular expression.
- – inactive / – active object filter.
- – filter list of objects by name or regular expression.
- – apply changes
- ▼ – drop-down menu with the directions of changes. If you click the DB label, you will also see the menu with the directions of changes.
- – get changes
- ▼ – drop-down menu with the databases sources. You can select the database sources from the right-click menu of the selected DB 127.0.0.1 on the editor tool bar (2).

The ▼ drop-down menus allow us to launch the operations of receiving and applying changes with overriding some settings.

Diff table shows the list of objects different for the DB schemas under comparison. Here you can find the info on object type, change type, object name, container, git user, database user.

Object type - the following object types are supported: SCHEMA, TYPE, SEQUENCE, TABLE, FUNCTION, PROCEDURE, VIEW, CONSTRAINT, INDEX, TRIGGER.

CAST, EXTENSION, DOMAIN, OPERATOR, FTS_PARSER, FTS_TEMPLATE, FTS_DICTIONARY, FTS_CONFIGURATION, AGGREGATE, RULE, POLICY, EVENT TRIGER are supported additionally for PostgreSQL.

USER, ROLE, ASSEMBLY are supported additionally for MS SQL.

Change type - an object can be in one of three conditions: exist only in a database, only in a project, or in both a project and a database. Depending on the condition and the direction of changes, objects in the list are marked as ‘delete’, ‘add’ and ‘edit’ or ‘CREATE’, ‘DROP’ and ‘ALTER’ respectively.

Container is the name of the parent object. For example, for an index, it is the name of a table or a view this index belongs to.

Git user - name of the user, who was the last one to change the project file for this object. In case if the files were changed locally, you’ll see a ‘*’ next to the username. To see this column, you should connect the project to the version control system and enable the corresponding option on the Project editor settings page.

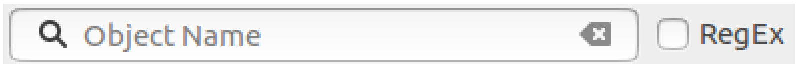
Database user - name of the database user who was the last one to change the object in the database. To display this column, you’ll need the pg_dbo_timestamp extension.

Comparison panel shows changes which took place in SQL view of an object.

Object counter shows the selected and the total number of objects in the table.

7.2 Object search in the differences table

The most convenient way of searching for the objects in the differences table is the search box above the changes list.

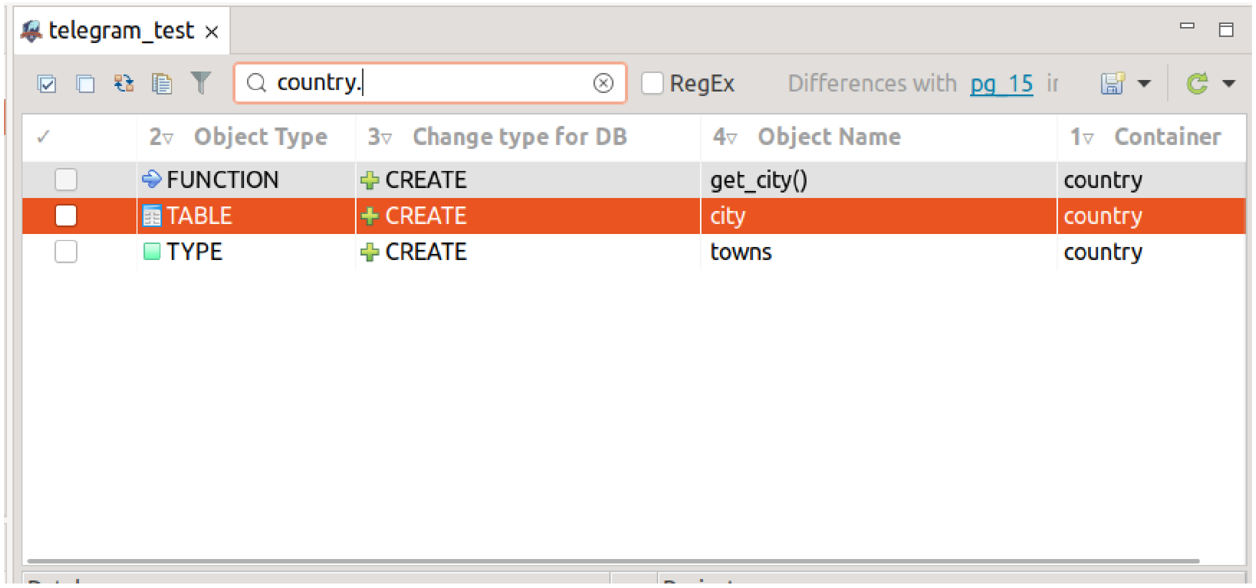


The search is conducted by the names of the objects. When RegEx is ticked off, the search is conducted with the use of regular expressions.

To start with, just enter a part of the name into the search box. The objects satisfying this condition will be displayed in the differences table.

The search box stores the last 200 entries.

To search by the qualified object name, enter the name of the schema with . in the search field.



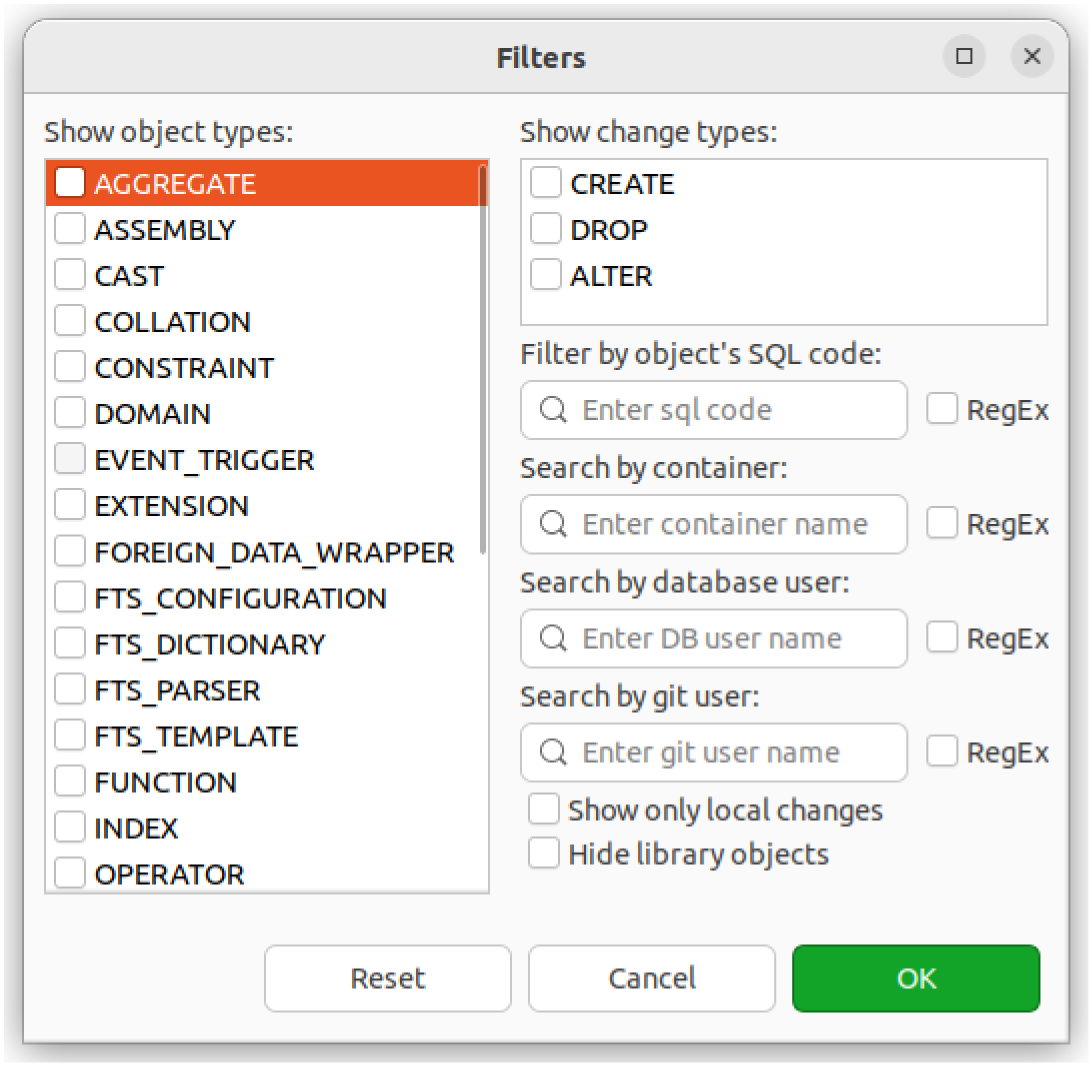
✓	2 Object Type	3 Change type for DB	4 Object Name	1 Container
☐	FUNCTION	CREATE	get_city()	country
☐	TABLE	CREATE	city	country
☐	TYPE	CREATE	towns	country

As a result, all the objects of this schema, excluding the schema itself, will be displayed in the differences table.

Enter its name to display the schema only.

7.3 Object filters

To filter the objects, you can use the dialog window which will appear after you click above the changes list.



The object type list allows to display only the selected types of objects. Selecting the TABLE or VIEW types will display all the descendant objects.

The changes type list allows to display only the selected types of changes.

Filter by object's SQL code searches for matches in the object generation code.

Search by container searches for the objects located in the schemas with appropriate names. Schema objects will be displayed as well.

Search by database user searches for matches in the authors of object changes in the thirdparty database. The pg_dbo_timestamp extension is necessary for work.

Search by git user searches for matches in the authors of the latest object change in local repository. This can be used if the project is under the version control system.

Show only local changes displays the changes that took place in local repository after the last commit. This can be used if the project is under the version control system.

Hide library objects hides the objects downloaded from libraries.

To apply the filters, click OK.

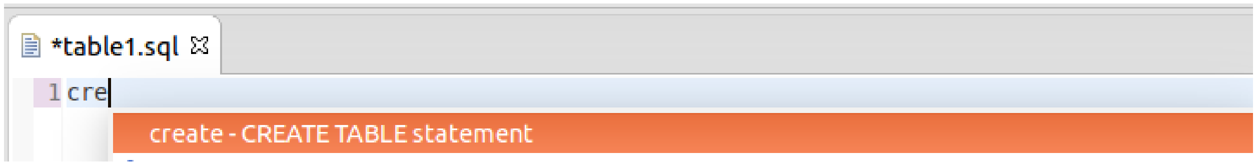
To cancel current changes, click Cancel.

To reset all the filters, click Reset.

When the filter is on, the button icon will change to .

7.4 SQL editor

SQL editor is not different from a regular Eclipse text editor appearance-wise, but it has functions of SQL syntax highlighting, SQL requests template autofilling, database schema objects navigation, and code formatting.



When using SQL-editor on the Eclipse tool bar, you will have the following options:

- Quick update. Execution of the migration script for differences between the object in the current file of the project and the respective object in the database. If the changes concern objects from other files or modify the data (for example, delete a column), there will be no update.

- Execute selection. Execution of the current migration script (or the highlighted text) in the selected database.

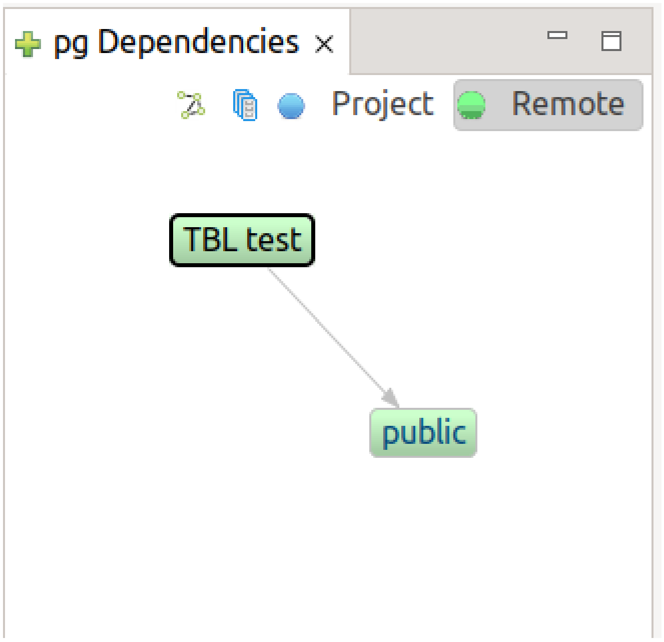
- Cancel execution. Cancel of execution of the current migration script.

- Get changes. Refreshes the editor of the project containing the current file. Applicable to project files only.

To format the highlighted piece of code, press Shift + Ctrl + F or select Format in the right-click menu.

7.5 Viewing the dependences of DB objects

The pg Dependencies view shows the dependencies containing the current object selected on the diff panel of the active project editor.

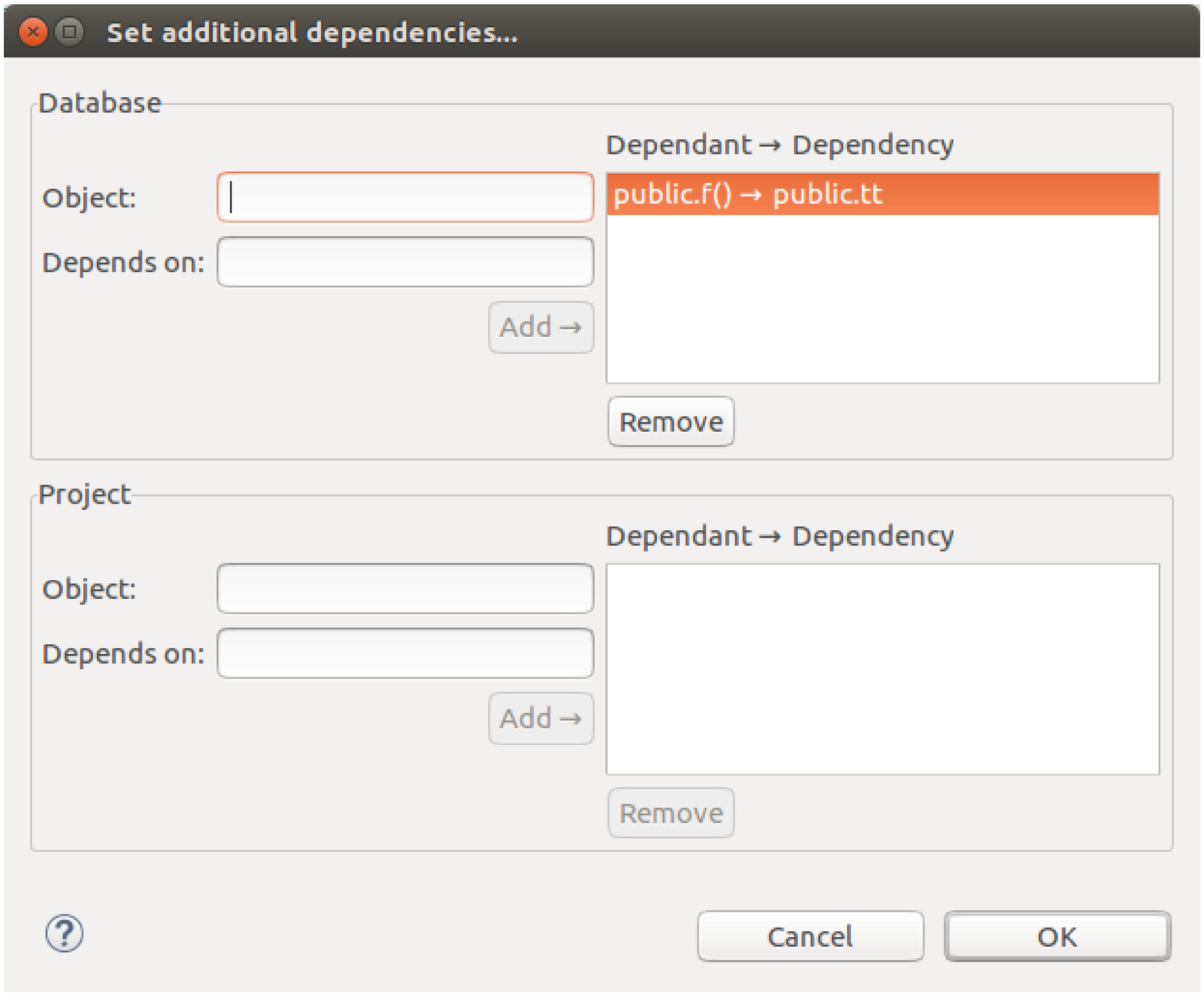


The arrows show the direction of connection of the dependant object towards its dependence.

The Project - Remote switch allows you to select the database under comparison for which you want to display objects and dependencies. After switching, you should select the element on the editor diff panel once again.

The Show columns button allows displaying the columns of the tables of the current objects as well as its dependencies.

Add depcy allows you to open the dialog of manual adding of the dependencies.



public.f() → public.tt

In this window, you can directly set the dependences between the DB objects. This may prove useful, for example, in case if automated recognition won't work for certain complex dependences. The added differences will be taken into consideration during generating the expression sequences for the migration script.

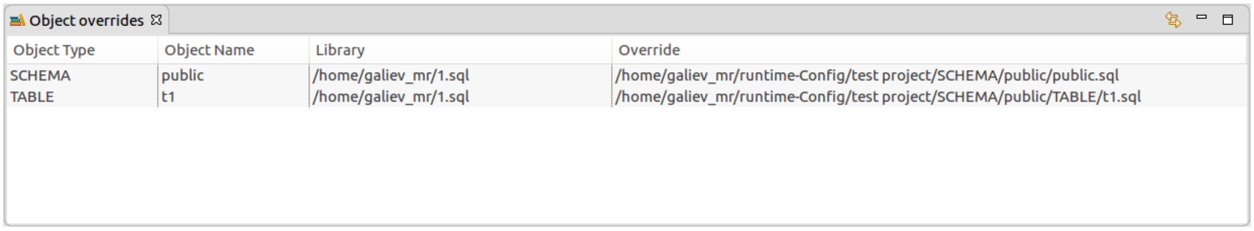
The window consists of two parts which serve for adding the dependences to the DB under comparison.

To add dependencies between objects, simply start entering the first letters of the object name, select the names of the dependent objects in the drop-down list and click Add. You'll see the dependency in the list of added.

To remove a chain of dependent objects, highlight it and click Remove.

7.6 Object overrides

The Object overrides view displays the list of overridden objects in libraries for the active project editor. This view is displayed automatically when receiving changes if there is at least one conflict present.



Object Type	Object Name	Library	Override
SCHEMA	public	/home/galiev_mr/1.sql	/home/galiev_mr/runtime-Config/test project/SCHEMA/public/public.sql
TABLE	t1	/home/galiev_mr/1.sql	/home/galiev_mr/runtime-Config/test project/SCHEMA/public/TABLE/t1.sql

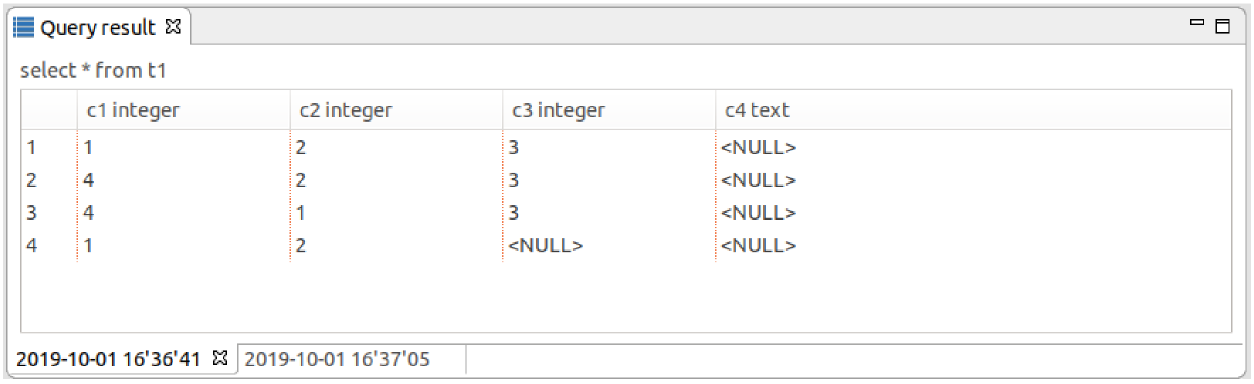
The button shows the notes concerning only the objects selected on the differences panel of the active project editor.

Right-click menu allows you to open both versions of the object, as well as to view the differences in a dedicated comparison editor.

7.7 Query result

Query result view displays query results. Each query is displayed in a separate tab.

Attention: Large samples may cause the graphical interface to "freeze".



select * from t1

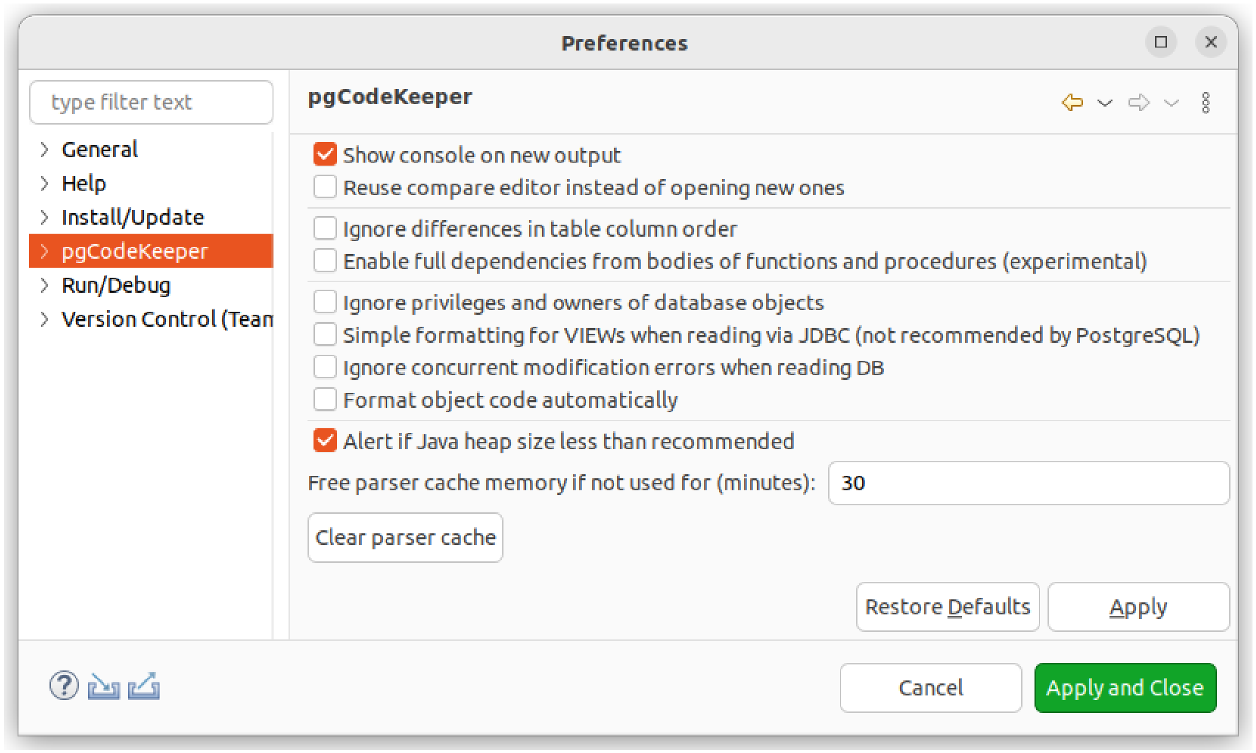
	c1 integer	c2 integer	c3 integer	c4 text
1	1	2	3	<NULL>
2	4	2	3	<NULL>
3	4	1	3	<NULL>
4	1	2	<NULL>	<NULL>

Глава

EIGHT

GLOBAL SETTINGS

8.1 pgCodeKeeper basic settings



pgCodeKeeper

- [✓] Show console on new output
- [ ] Reuse compare editor instead of opening new ones
- [ ] Ignore differences in table column order
- [ ] Enable full dependencies from bodies of functions and procedures (experimental)
- [ ] Ignore privileges and owners of database objects
- [ ] Simple formatting for VIEWs when reading via JDBC (not recommended by PostgreSQL)
- [ ] Ignore concurrent modification errors when reading DB
- [ ] Format object code automatically
- [✓] Alert if Java heap size less than recommended

Free parser cache memory if not used for (minutes): 30

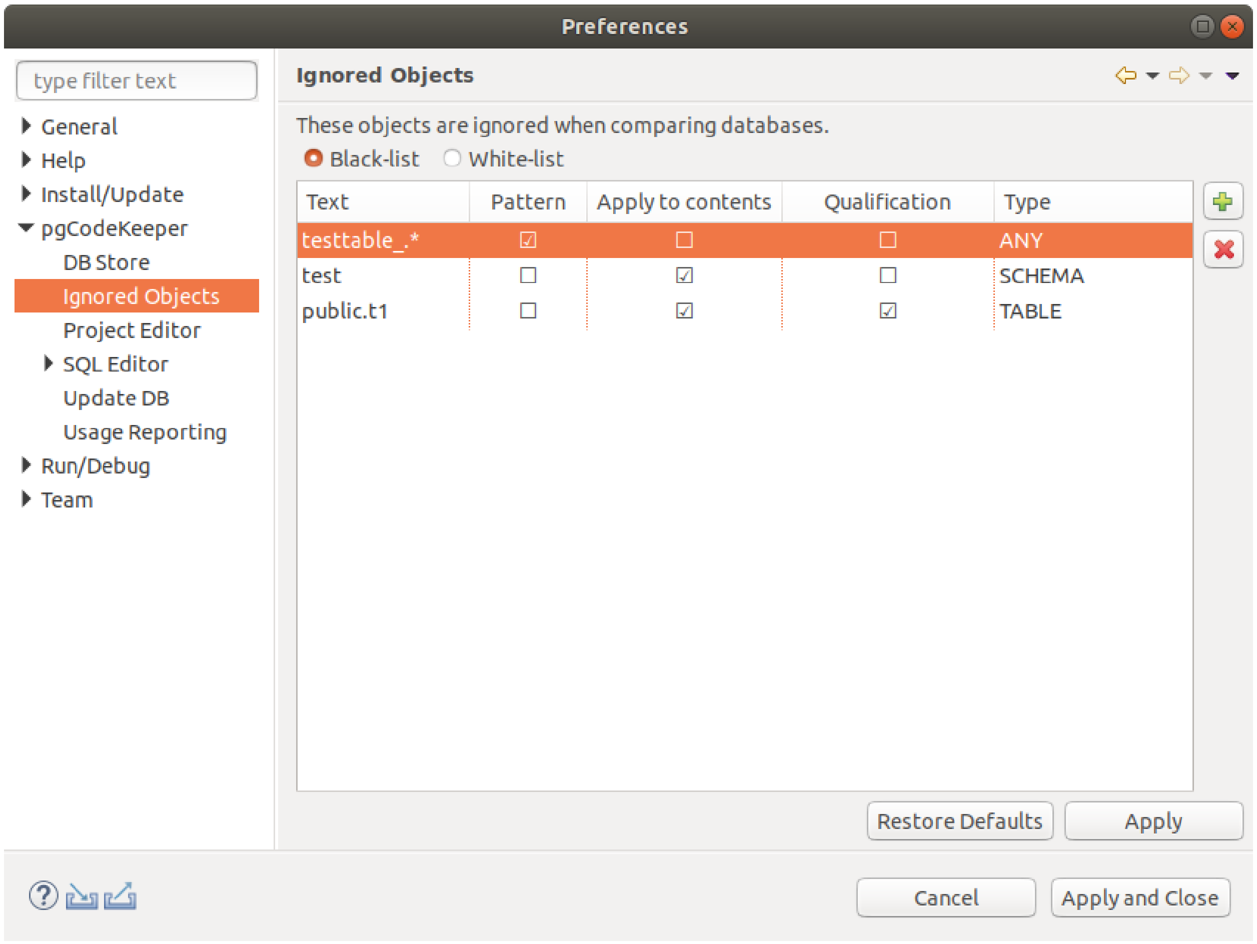
- Show console on new output - allows to automatically show program console when opening it.
- Ignore privileges and owners of database objects - allows to disable the search for differences in the properties of the objects related to DB roles.
- Ignore differences in table column order - allows to ignore the order of colums when comparing the tables.
- Reuse compare editor instead of opening new ones - allows using the already running comparison editor for comparing database objects (Select Show Diff in the right-click menu of the differences table).
- Enable full dependencies from bodies of functions and procedures (experimental) - allows searching the dependences on functions and procedures within the bodies of other functions and procedures.
- Simple formatting for VIEWs when reading via JDBC (not recommended by PostgreSQL) - allows using simplified view editing, where all unnecessary brackets are removed from the expression. This

format might not be supported in the further PostgeSQL versions.

- Ignore concurrent modification errors when reading DB - allows you to ignore errors that appear when modifying a DB object while reading it.
- Format object code automatically allows display and save the formatted function code in plpgsql and sql according to the SQL Editor formatting settings. For more info about the code formatting settings, see Formatting.
- Free parser cache memory if not used for (minutes): - allows you to automatically clear parser cache not used for a certain amount of time. Setting this to 0 disables this behavior.
- Clear parser cache - allows to clear parser cache.

8.2 Excluded objects

You can specify the objects that should not be considered when comparing databases on the pgCodeKeeper -> Ignored objects settings page.



These objects are ignored when comparing databases.

[✓ Black-list] White-list

Text	Pattern	Apply to contents	Qualification	Type
testtable_.*	☑	☐	☐	ANY
test	☐	☑	☐	SCHEMA
public.t1	☐	☑	☑	TABLE

The Black list - White list switch allows to invert black list to white list and the other way around.

- Text – name of the object.
- Pattern – consider the name of the object as regular expression.
- Apply to contents – apply the rule for the matched object and all of its content.

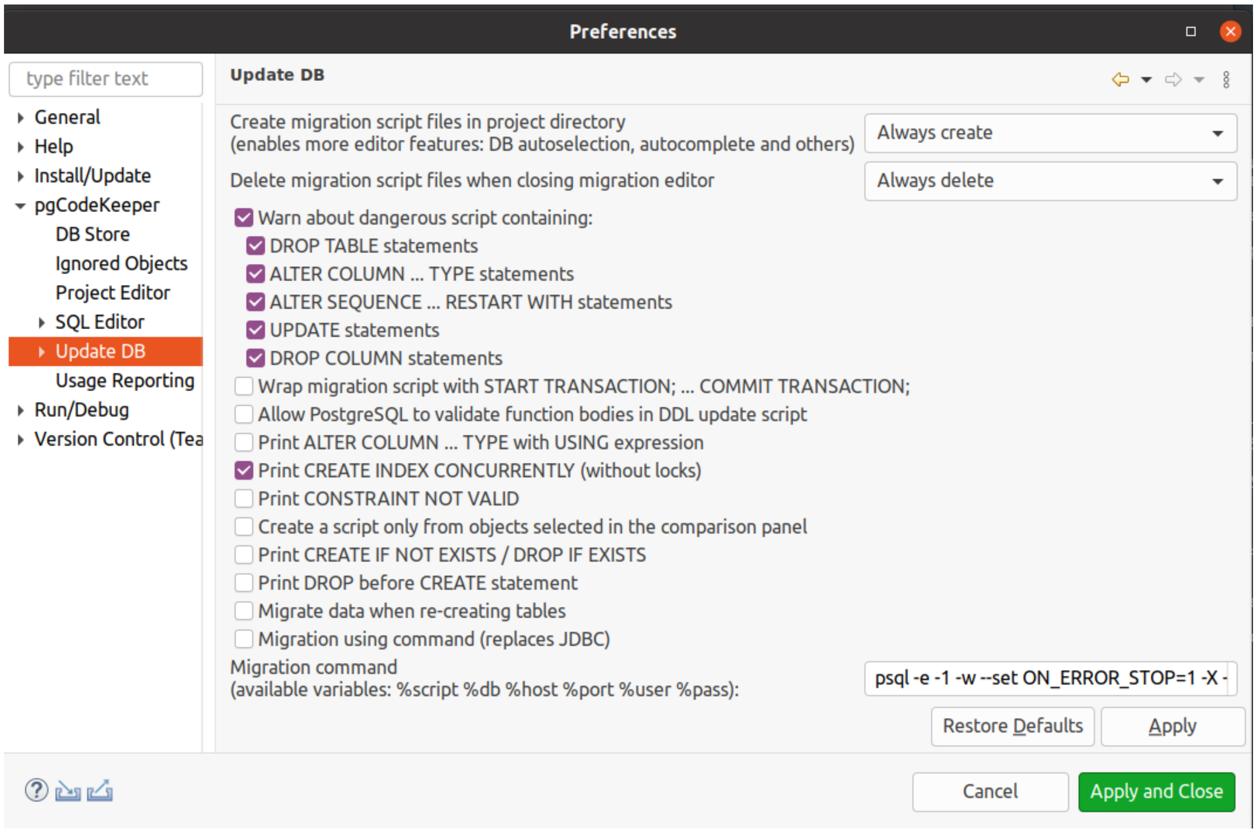
- Qualification – search objects by qualified name.
- Type - object type ("ANY" is used for specifying any object type).

To add or remove an object to the list of excluded, use the ➕ and ✖ buttons respectively.

Note: In this case, the list of excluded objects will be general and will expand its effect at all the projects present in the projects directory. A detailed description of work with lists see in Ignore List.

8.3 DB update

Settings managing the database updates.



Preferences

Update DB

Create migration script files in project directory (enables more editor features: DB autoselection, autocomplete and others): Always create

Delete migration script files when closing migration editor: Always delete

- [✓] Warn about dangerous script containing:
  - [✓] DROP TABLE statements
  - [✓] ALTER COLUMN ... TYPE statements
  - [✓] ALTER SEQUENCE ... RESTART WITH statements
  - [✓] UPDATE statements
  - [✓] DROP COLUMN statements
- [ ] Wrap migration script with START TRANSACTION; ... COMMIT TRANSACTION;
- [ ] Allow PostgreSQL to validate function bodies in DDL update script
- [ ] Print ALTER COLUMN ... TYPE with USING expression
- [✓] Print CREATE INDEX CONCURRENTLY (without locks)
- [ ] Print CONSTRAINT NOT VALID
- [ ] Create a script only from objects selected in the comparison panel
- [ ] Print CREATE IF NOT EXISTS / DROP IF EXISTS
- [ ] Print DROP before CREATE statement
- [ ] Migrate data when re-creating tables
- [ ] Migration using command (replaces JDBC)

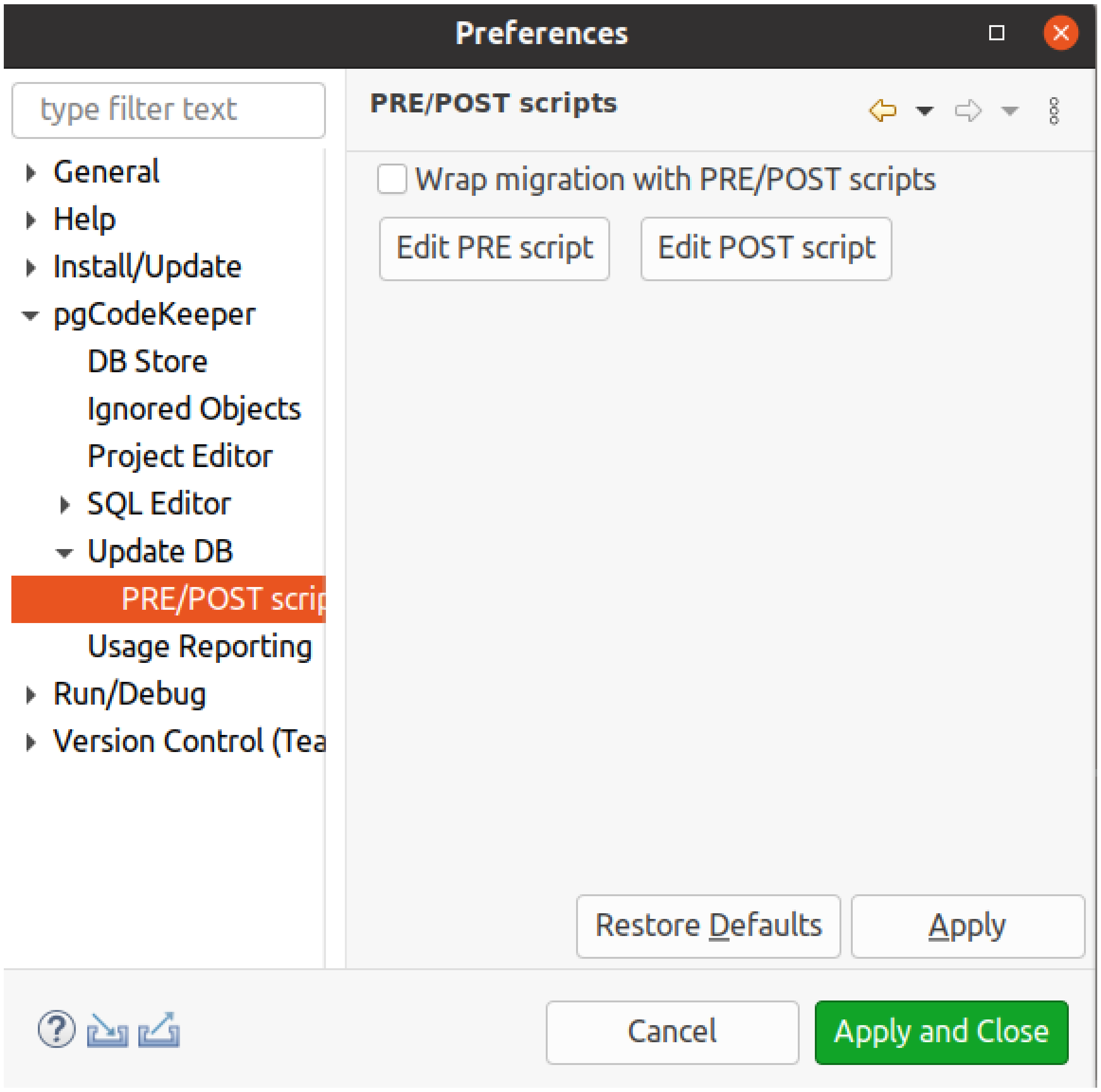
Migration command (available variables: %script %db %host %port %user %pass): psql -e -1 -w --set ON_ERROR_STOP=1 -X -

On the pgCodeKeeper -> Update DB settings page you can adjust parameters for the work of the migration script editor.

- Create migration script files in project directory – sets operations when creating migration scripts.
- Delete migration script files when closing migration editor – sets oprations after closing the migration scripts, if they were created in the project directory.
- Warn about dangerous scripts containing: - allows to show notifications if any selected dangerous expressions were formed during script generation.
- Show script output in separate window – allows to view the message with the server response when migration is complete.

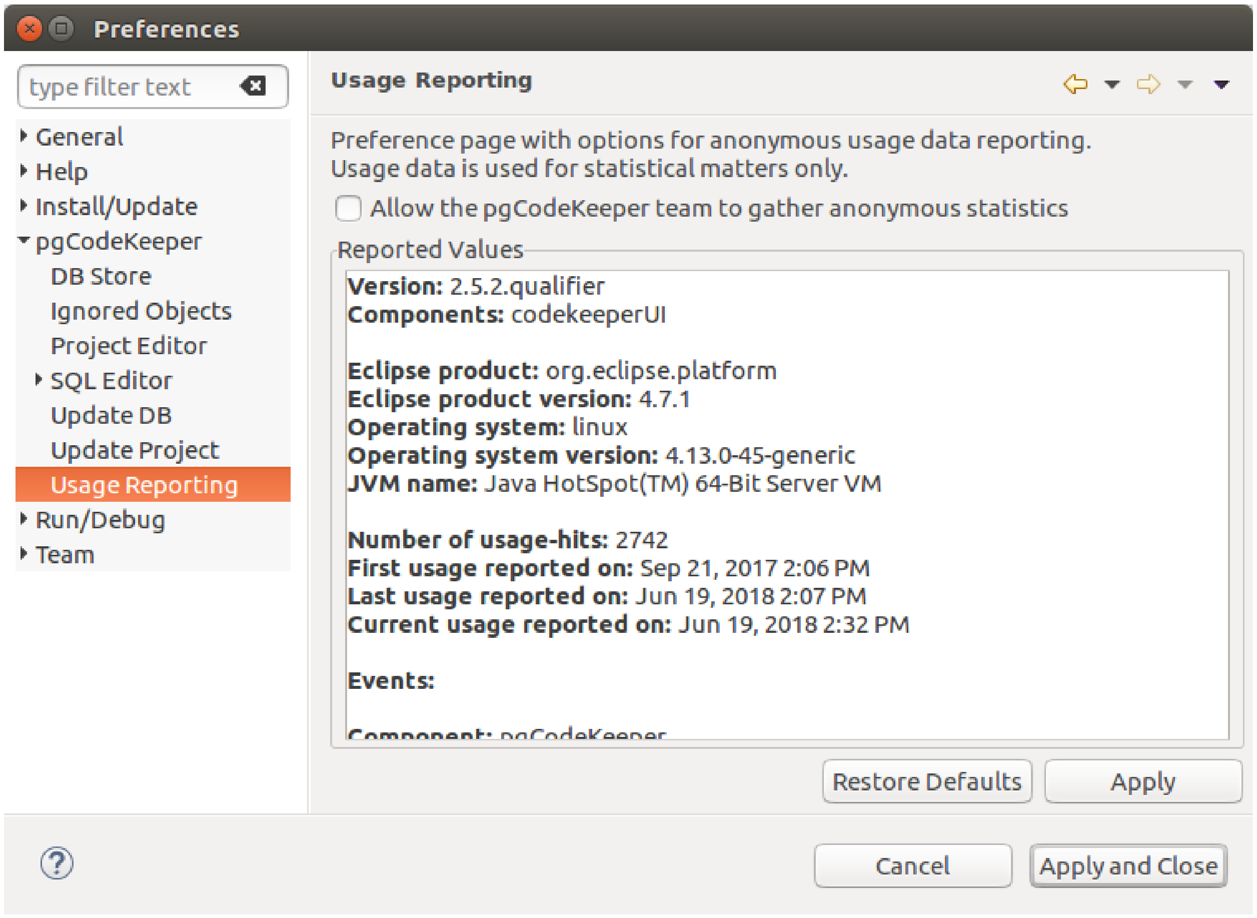
- Surround migration script with START TRANSACTION; … COMMIT TRANSACTION; – allows to add the start … commit pair to the generating scripts.
- Allow PostgreSQL to validate function bodies in DDL update script – allows to add the SET check_function_bodies = true; check in the beginning of a generating script.
- Print ALTER COLUMN … TYPE … with USING expression – allows to add the expression, permitting to change the data type when changing the column type, to the script
- Print CREATE INDEX CONCURRENTLY (without locks) – allows creating indexes in the CONCURRENTLY mode in scripts.
- Print CONSTRAINT NOT VALID - allows adding the NOT VALID statement to the migration script for constraints.
- Print CREATE IF NOT EXISTS/DROP IF EXISTS allows to add the IF NOT EXISTS/IF EXISTS statement to the migration script in the object's CREATE / DROP statement.
- Print DROP before CREATE statement allows to add the DROP statement before the CREATE statement in the migration script.
- Create a script only from objects selected in the comparison panel – allows to exclude the objects, which were not selected explicitly, from the script.
- Migrate data when re-creating tables allows to save the data when recreating a table. When recreating, the existing table is renamed and a new table is created. The data from the original table is transferred to the new one. The old, renamed table is deleted. The IDENTITY SEQUENCE status is transferred from the old table to the new one. Regular SEQUENCE are not supported.
- Migration command - allows to use another utility or bootup options when applying migration script to the database.

On the Update DB -> PRE/POST script tab, there are settings for adding the PRE/POST scripts to migrations. In the project, it is possible to create the PRE and POST directories, content of which will be added to the beginning and the end of the main script. Also, the PRE/POST scripts are available in global settings and in the CLI settings. To edit the content of the PRE/POST scripts, select the corresponding buttons.



8.4 Usage reports

Settings managing the usage reports.



Usage Reporting

Preference page with options for anonymous usage data reporting.
Usage data is used for statistical matters only.

☐ Allow the pgCodeKeeper team to gather anonymous statistics

Reported Values

Version: 2.5.2.qualifier
Components: codekeeperUI

Eclipse product: org.eclipse.platform
Eclipse product version: 4.7.1
Operating system: linux
Operating system version: 4.13.0-45-generic
JVM name: Java HotSpot(TM) 64-Bit Server VM

Number of usage-hits: 2742
First usage reported on: Sep 21, 2017 2:06 PM
Last usage reported on: Jun 19, 2018 2:07 PM
Current usage reported on: Jun 19, 2018 2:32 PM

Events:

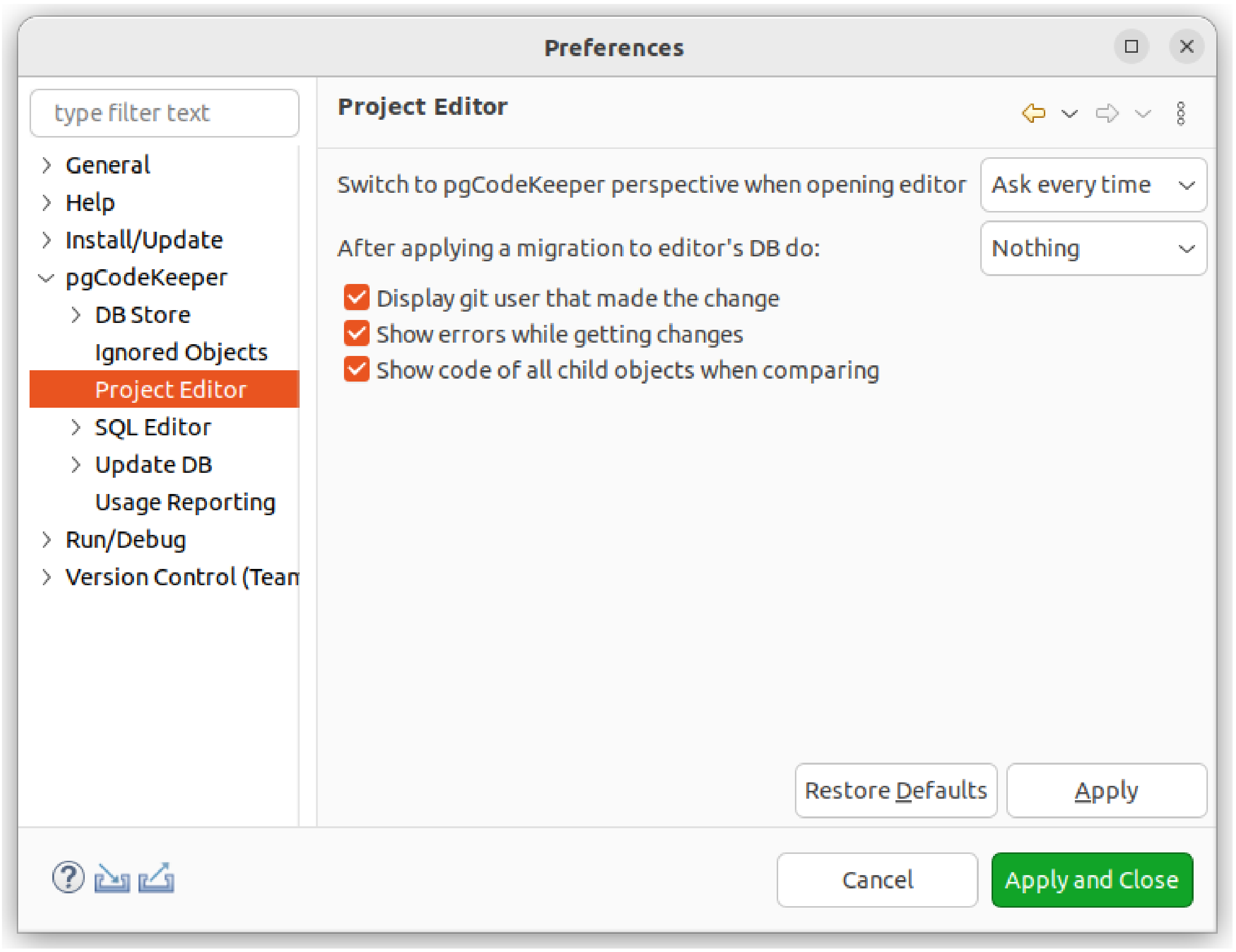
Component: pgCodeKeeper

On the pgCodeKeeper -> Usage Reporting settings page you can enable or disable sending the pgCodeKeeper usage statistics.

The values which are sent for the statistics you can review in the Reported Values field.

8.5 Project editor

Settings managing the project editor.



Project Editor

Switch to pgCodeKeeper perspective when opening editor: Ask every time

After applying a migration to editor's DB do: Nothing

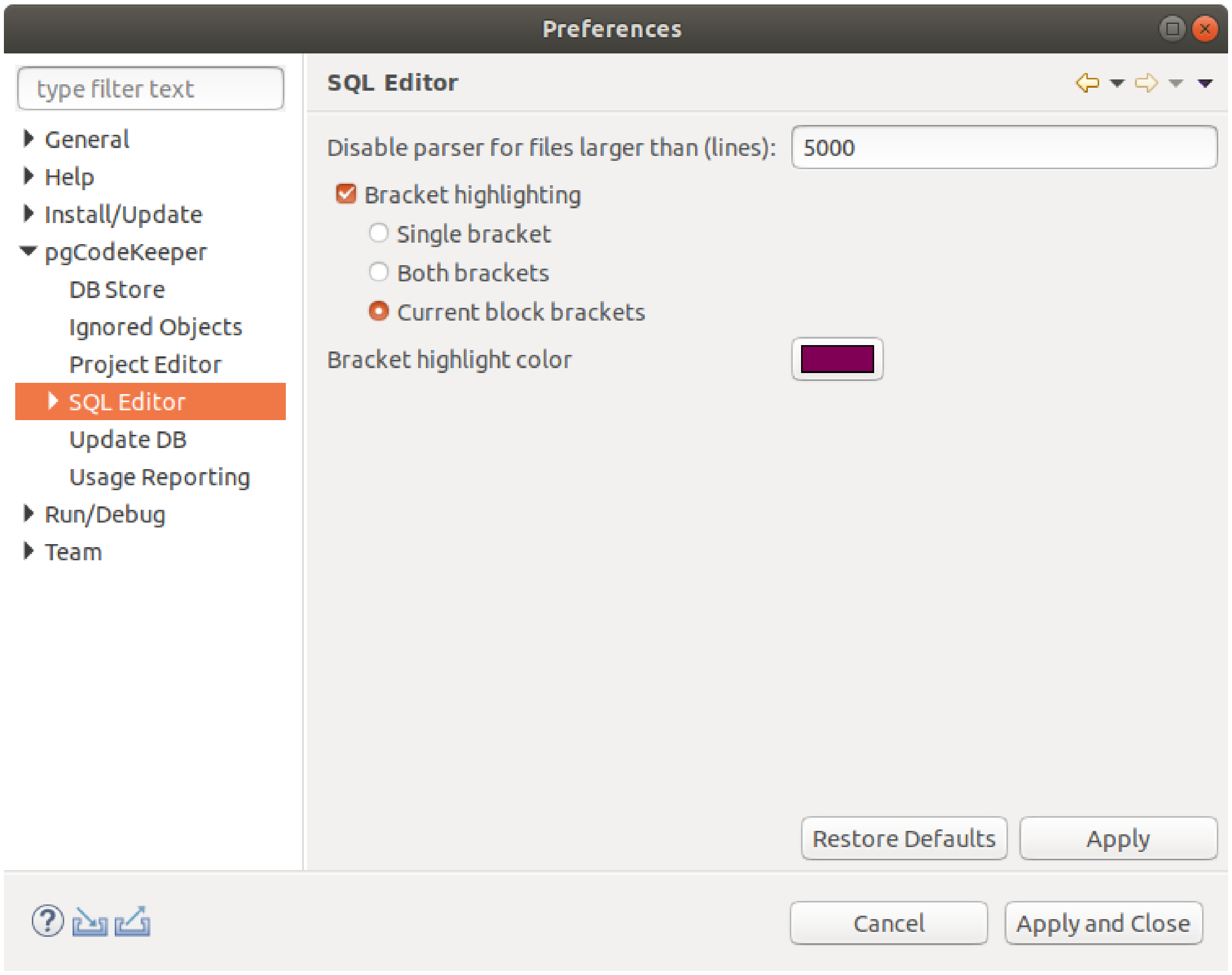
- [x] Display git user that made the change
- [x] Show errors while getting changes
- [x] Show code of all child objects when comparing

On the pgCodeKeeper -> Project editor settings page you can adjust the parameters of the project editor.

- Switch to pgCodeKeeper perspective when opening editor – sets operations for switching the perspective while starting the project editor.
- After applying a migration to editor's DB do: – sets the editor operations while implementing the migration scripts.
- Display git user that math the change – allows to display the column, containing the git user who made the latest change to the file, in the editor. The project should be under the version control system.
- Show errors while getting changes allows displaying the list of errors discovered while getting changes.
- Show code of all child objects when comparing - allows to display the code of all child objects on the comparison panel, even if there are no differences.

8.6 SQL Editor

Settings managing the SQL editor.



SQL Editor

Disable parser for files larger than (lines): 5000

[✓] Bracket highlighting

( ) Single bracket

( ) Both brackets

(● ✓) Current block brackets

Bracket highlight color

On the pgCodeKeeper -> SQL Editor settings page you can adjust the options of the SQL editor.

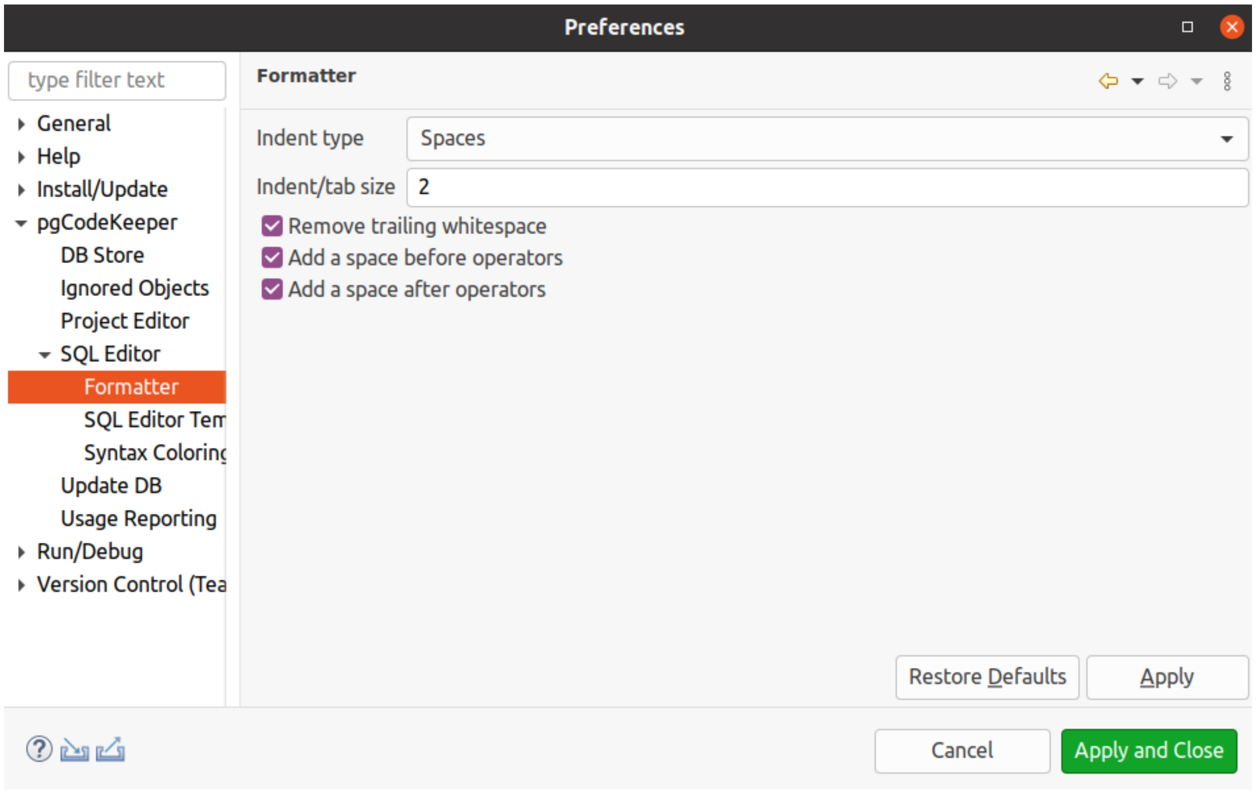
- Disable parser for files larger than (lines): – allows you to select a size of files for which the pgCodeKeeper builder is disabled. This speeds up the process of opening the file and makes your work with it faster in general, but it also disables the search for errors and links to objects within this file. Setting this to 0 disables this behavior.
- Bracket highlight – enables bracket highlight in one of the modes.
- Single bracket – when selecting a bracket, highlights the matching bracket.
- Both brackets – when selecting a bracket, highlights the selected and the matching brackets.
- Current block brackets – highlights the brackets between which the cursor is situated.
- Bracket highlight color – allows to select the color of brackets highlighting.

8.7 Formatting

Settings for managing the formatting rules.

On the pgCodeKeeper -> SQL Editor -> Formatting settings page, you can:

- set an indent type applied at the beginning of code lines;
- set an indent width for a selected indent type;
- enable/disable the removal of spaces at the end of lines;
- set the addition of spaces before/after operators.



Formatter

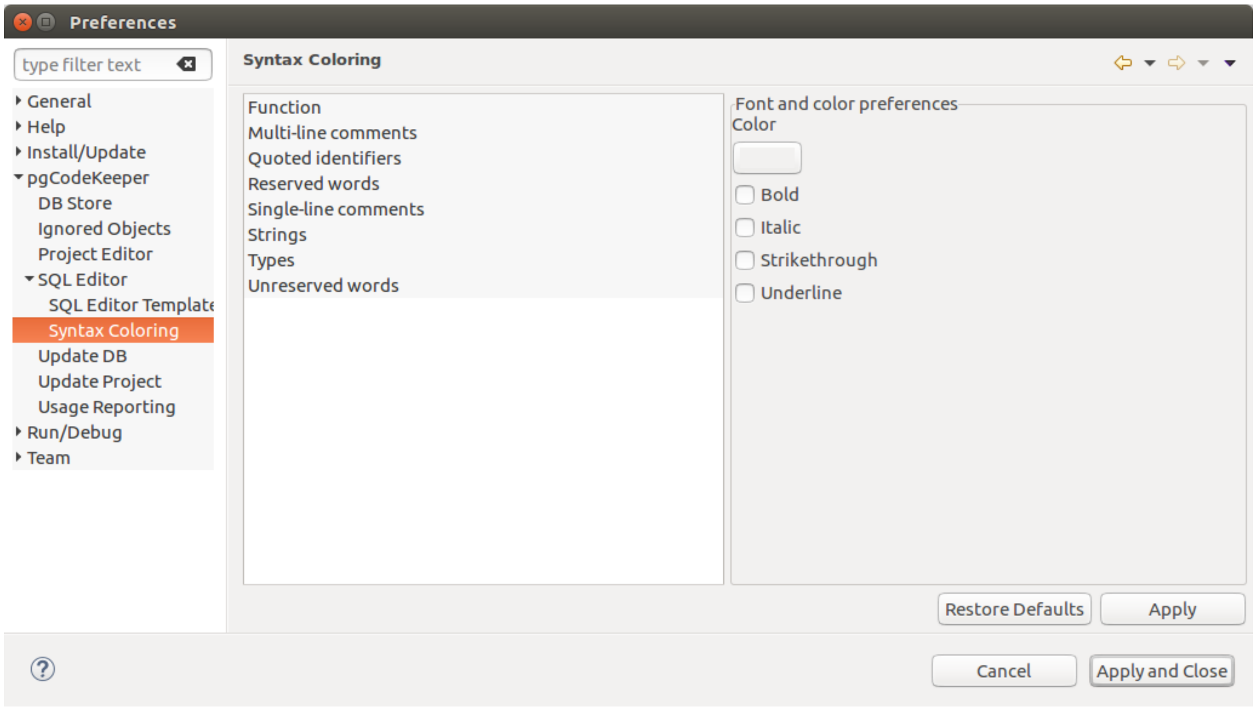
Indent type: Spaces

Indent/tab size: 2

- [✓] Remove trailing whitespace
- [✓] Add a space before operators
- [✓] Add a space after operators

8.8 Syntax Coloring

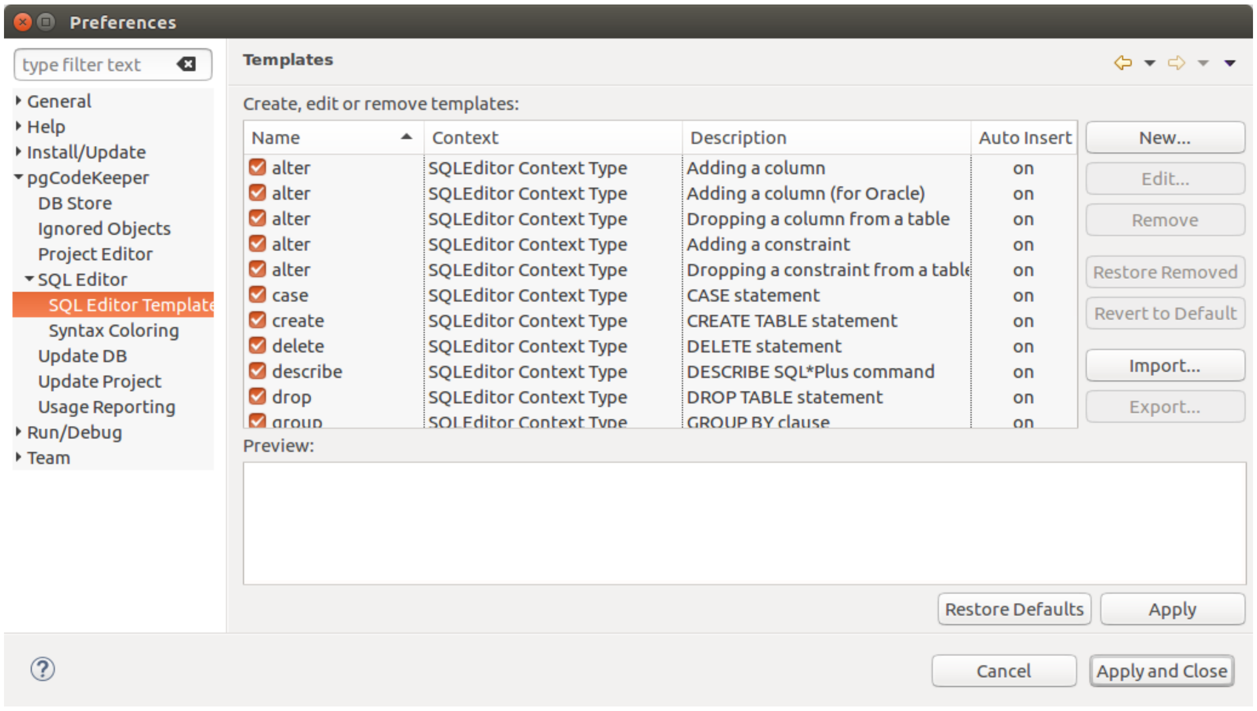
Settings managing the syntax highlighting.



On the pgCodeKeeper -> SQL Editor -> Syntax highlighting you can set color and style for the font in which all the corresponding syntax will be displayed in the pgCodeKeeper SQL editor.

8.9 SQL Editor Templates

Creating, deleting and editing the templates for auto expand in the SQL editor



Preferences

type filter text

- General
- Help
- Install/Update
- pgCodeKeeper
  - DB Store
  - Ignored Objects
  - Project Editor
  - SQL Editor
    - SQL Editor Template
    - Syntax Coloring
  - Update DB
  - Update Project
  - Usage Reporting
- Run/Debug
- Team

Templates

Create, edit or remove templates:

Name	Context	Description	Auto Insert
✓ alter	SQLEditor Context Type	Adding a column	on
✓ alter	SQLEditor Context Type	Adding a column (for Oracle)	on
✓ alter	SQLEditor Context Type	Dropping a column from a table	on
✓ alter	SQLEditor Context Type	Adding a constraint	on
✓ alter	SQLEditor Context Type	Dropping a constraint from a table	on
✓ case	SQLEditor Context Type	CASE statement	on
✓ create	SQLEditor Context Type	CREATE TABLE statement	on
✓ delete	SQLEditor Context Type	DELETE statement	on
✓ describe	SQLEditor Context Type	DESCRIBE SQL*Plus command	on
✓ drop	SQLEditor Context Type	DROP TABLE statement	on
✓ group	SQLEditor Context Type	GROUP BY clause	on

New... Edit... Remove Restore Removed Revert to Default Import... Export...

Preview:

Restore Defaults Apply

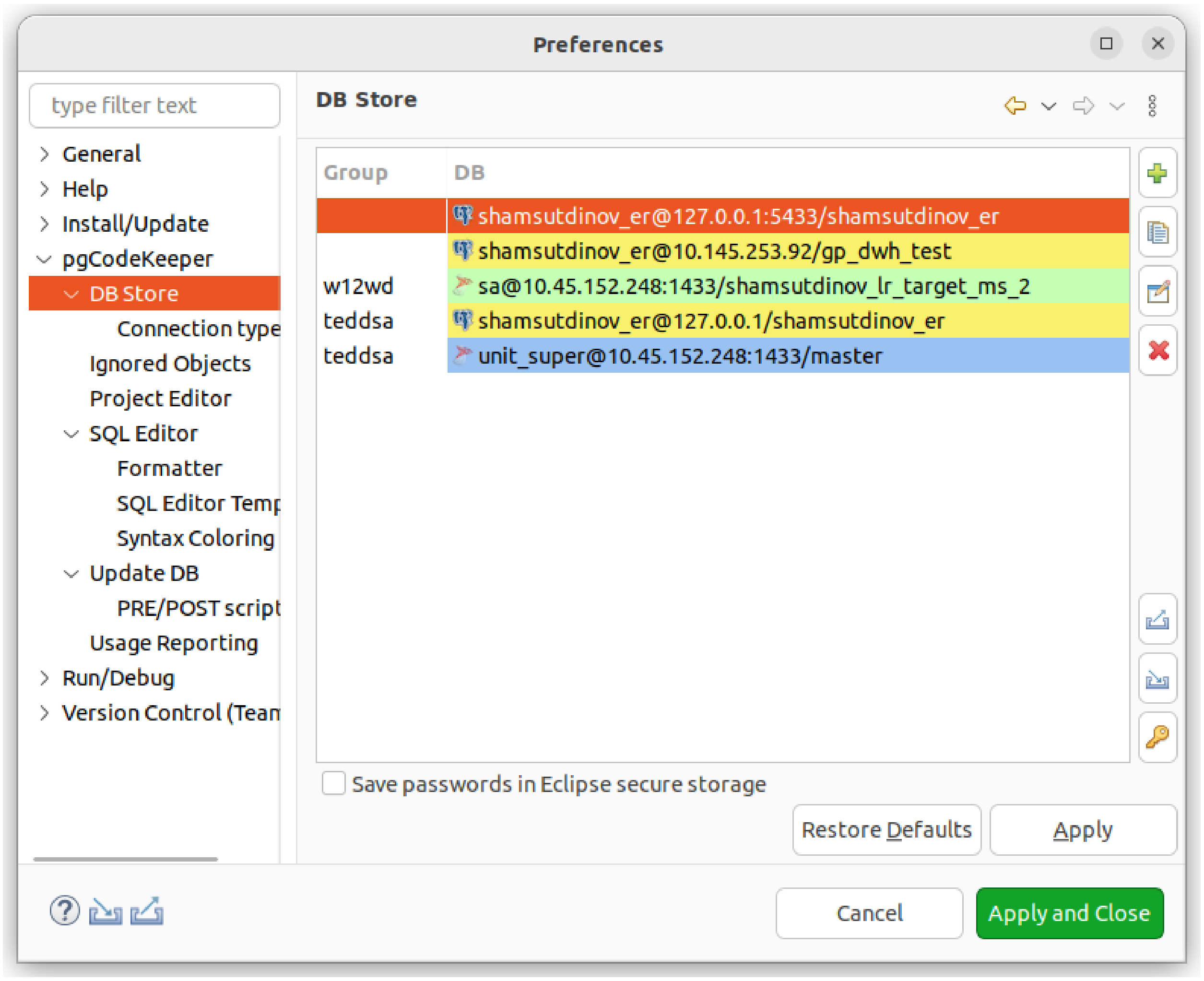
Cancel Apply and Close

On the pgCodeKeeper -> SQL Editor -> SQL Editor Templates settings page you can manage SQL templates, available in the pgCodeKeeper SQL editor. The list of available actions is displayed to the right of the templates list.

8.10 DB storage

Settings managing the storage of the parameters of the DB connection.

On the pgCodeKeeper -> DB Store settings page, there is a list of entries for connection to databases. To upload the connections list from a file, click Import connection list and select the required file containing the connections. You can also save a connections list to a separate file. To do that, click Export connection list.



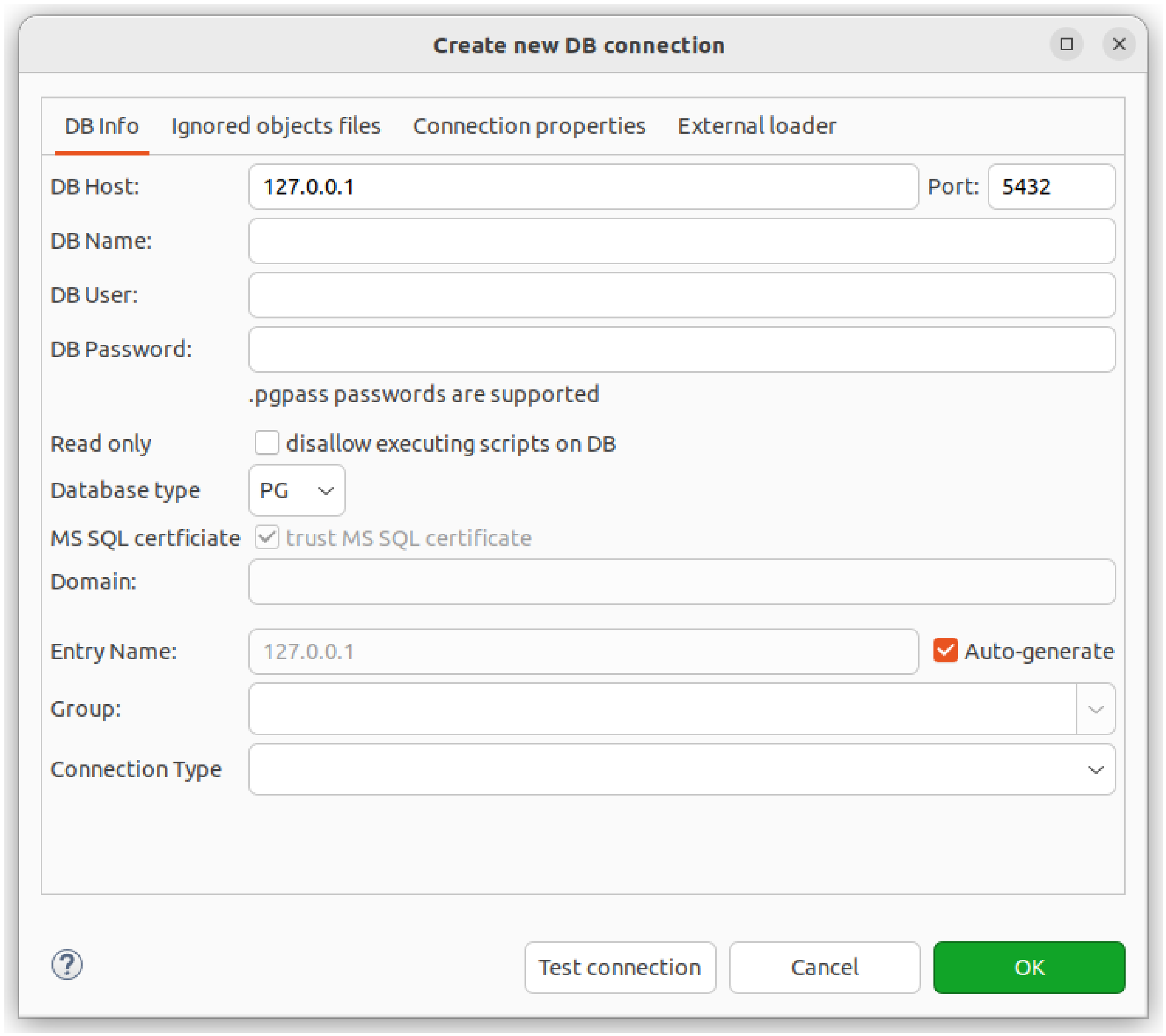
DB Store

Group	DB
	shamsutdinov_er@127.0.0.1:5433/shamsutdinov_er
	shamsutdinov_er@10.145.253.92/gp_dwh_test
w12wd	sa@10.45.152.248:1433/shamsutdinov_lr_target_ms_2
teddsa	shamsutdinov_er@127.0.0.1/shamsutdinov_er
teddsa	unit_super@10.45.152.248:1433/master

☐ Save passwords in Eclipse secure storage

Click ➕ to add a storage. The DB credentials dialog will appear, in which you should specify the parameters of the connection data: host, port, DB name, user, password, DB group, connection type, as well as forbid recording to DB and select the DB type (PostgreSQL or MS SQL). For MS SQL, you can specify a domain. If the MS SQL database type is selected, the trust MS SQL certificate, parameter will be set by default. This parameter can be disabled. You can specify the name of the record manually. To do this, disable the Auto-generate option.

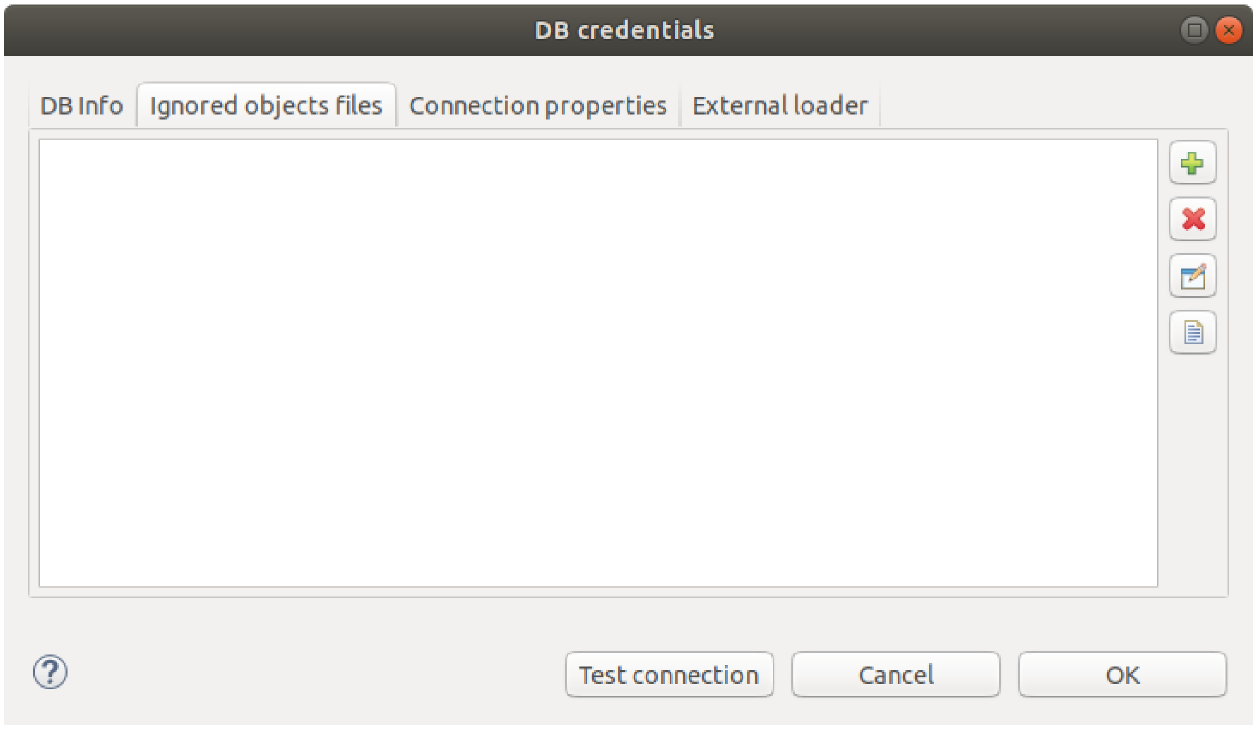
Note: pgCodeKeeper supports the pgpass file. To do this, leave the password field blank.


Create new DB connection

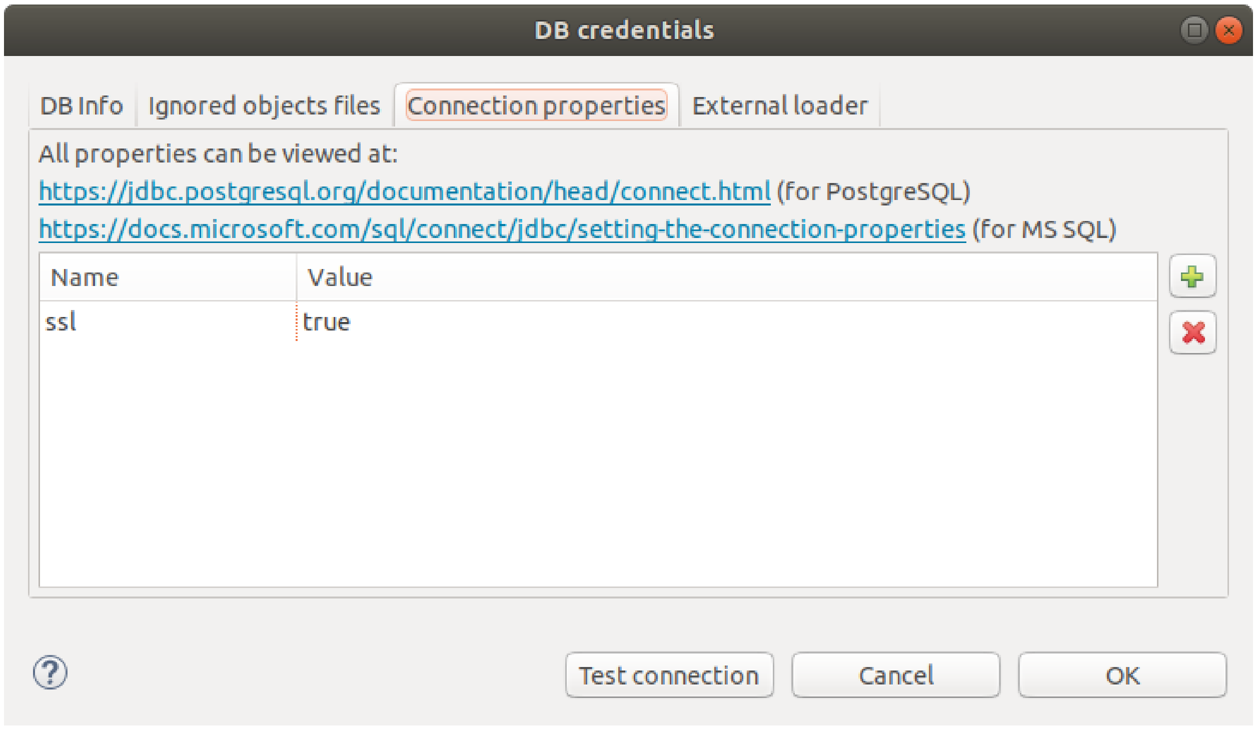
DB Info | Ignored objects files | Connection properties | External loader

DB Host:	127.0.0.1 Port: 5432
DB Name:	
DB User:	
DB Password:	
	.pgpass passwords are supported
Read only	☐ disallow executing scripts on DB
Database type	PG
MS SQL certficiate	✓ trust MS SQL certificate
Domain:	
Entry Name:	127.0.0.1 ✓ Auto-generate
Group:	
Connection Type	

Ignore List can be connected as an external file.



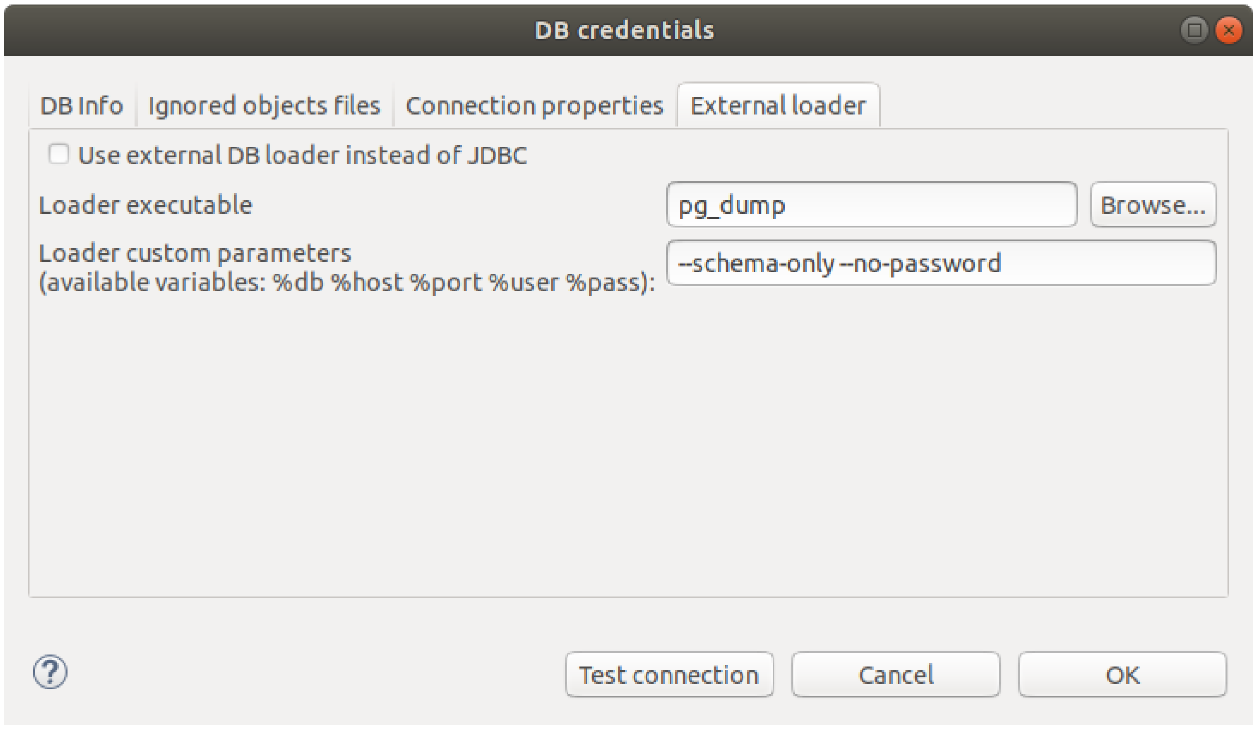
You can extend connection properties with the settings specified at:
https://jdbc.postgresql.org/documentation/head/connect.html (For PostgreSQL)
https://docs.microsoft.com/ru-ru/sql/connect/jdbc/setting-the-connection-properties (For MS SQL)



All properties can be viewed at:
https://jdbc.postgresql.org/documentation/head/connect.html (for PostgreSQL)
https://docs.microsoft.com/sql/connect/jdbc/setting-the-connection-properties (for MS SQL)

Name	Value
ssl	true

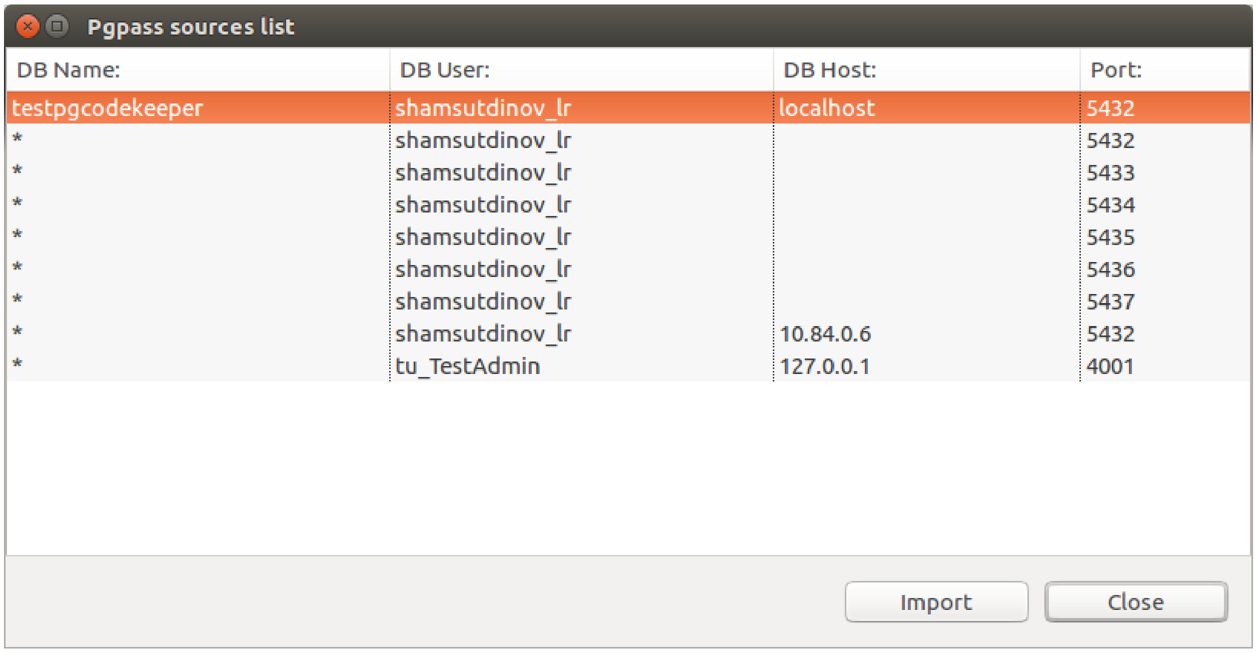
- Use external DB loader instead of JDBC - allows using external database loader instead of JDBC.
- Loader executable - path to a utility, executable file or a script used for loading.
- Loader custom parameters - utility activation custom parameters.



- [ ] Use external DB loader instead of JDBC
- Loader executable: pg_dump
- Loader custom parameters (available variables: %db %host %port %user %pass): --schema-only --no-password

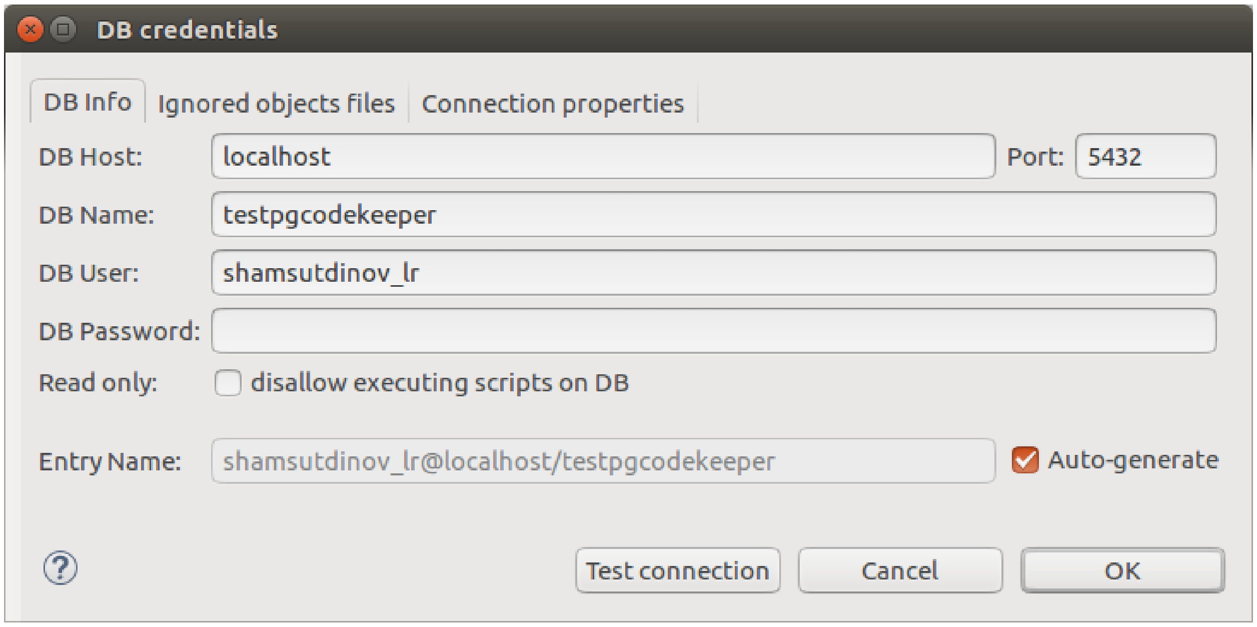
To add a storage basing on data from an already created storage, you should highlight the database storage object and click|copy|. The DB credentials dialogue will appear, in which you can change the connection parameters.

To add a storage basing on the data from the .pgpass file, click|pg_pass|. A dialogue will appear, in which you should select the .pgpass file. After that, the Pgpass sources list dialogue will open:



DB Name:	DB User:	DB Host:	Port:
testpgcodekeeper	shamsutdinov_lr	localhost	5432
*	shamsutdinov_lr		5432
*	shamsutdinov_lr		5433
*	shamsutdinov_lr		5434
*	shamsutdinov_lr		5435
*	shamsutdinov_lr		5436
*	shamsutdinov_lr		5437
*	shamsutdinov_lr	10.84.0.6	5432
*	tu_TestAdmin	127.0.0.1	4001

There, you should select the line with the data, basing on which the storage will be created. The DB credentials dialogue will appear, where you can confirm the connection parameters.



DB Host: localhost
Port: 5432
DB Name: testpgcodekeeper
DB User: shamsutdinov_lr
DB Password:
Read only: ☐ disallow executing scripts on DB
Entry Name: shamsutdinov_lr@localhost/testpgcodekeeper
☑ Auto-generate

After that, you can close the Pgpass sources list dialogue.

To edit a storage, highlight the database storage object and click 📝. The DB credentials dialogue will appear in which you can change the connection parameters.

To delete a storage, select the required record and click ✖.

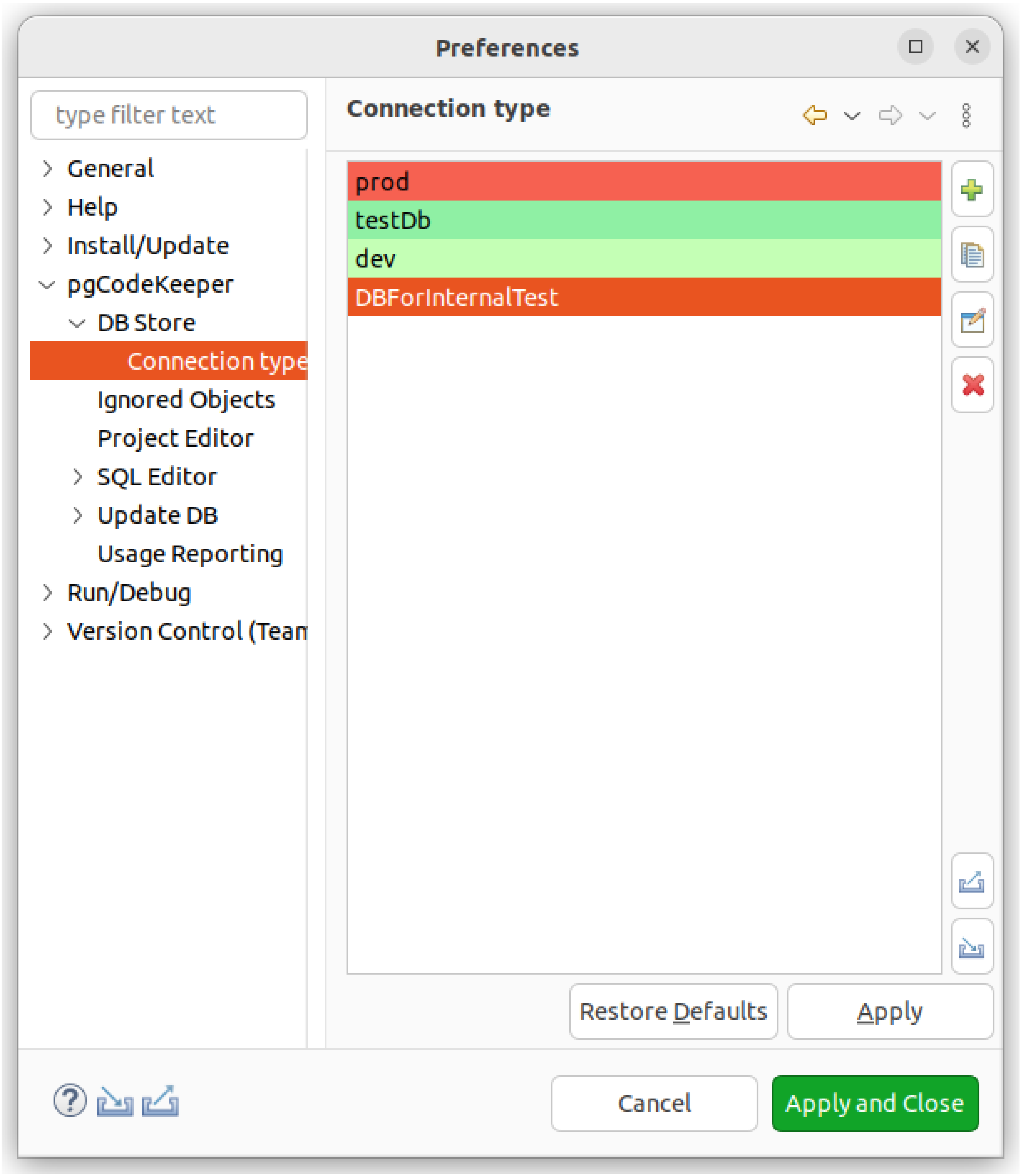
Attention: The Restore Defaults button on the settings page deletes all the records for the databases connection and creates one 'default' record without any data on the connection.

Attention: To save the changes, you should click Apply or Apply and Close in the DB connection parameters storage, on the settings page.

8.11 Connection type

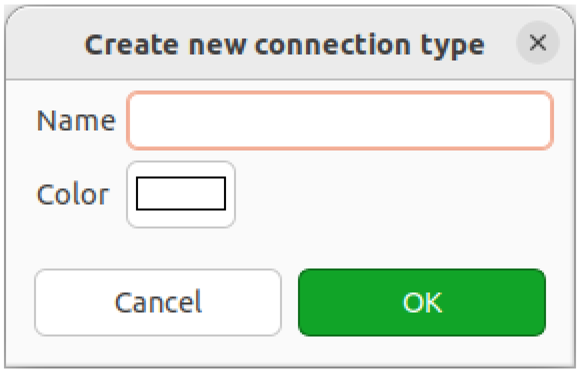
Settings configuring the connection type.

You can find the connection types list in pgCodeKeeper -> DB Store -> Connection type.

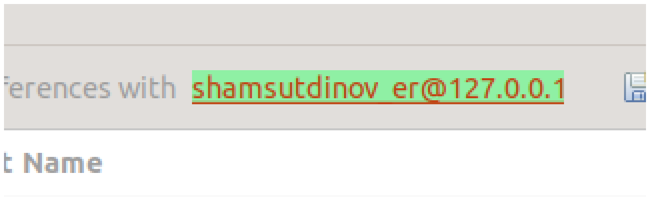


prod
testDb
dev
DBForInternalTest

To upload the list from a file, click Import and select the required file. You can also export the list to a separate file by clicking Export. To add a connection type, click . The Create new connection type dialog window will appear



- Name is the name of a connection type displayed on the list in the drop-down list of the DB credentials dialog window DB Store.
- Color is the color used to highlight the connection type on the DB storage list, on the toolbar and in the SQL editor



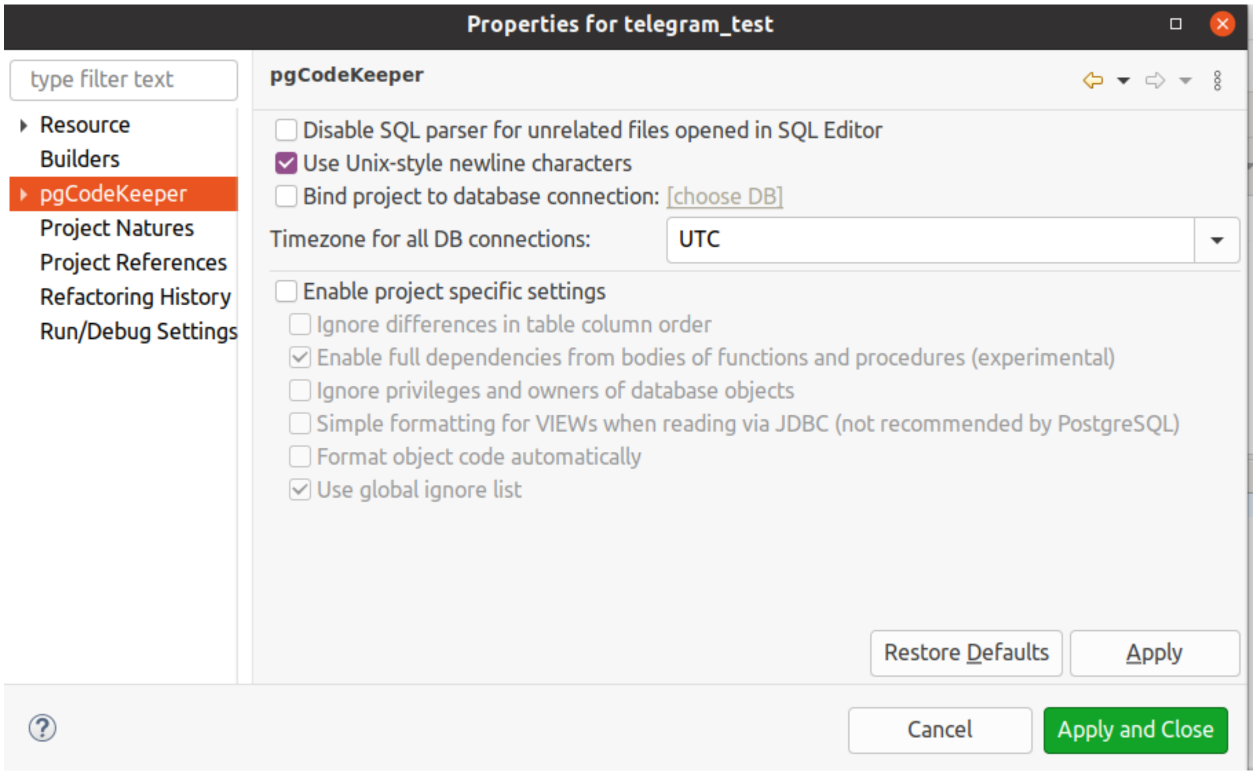
Глава

NINE

PROJECT SETTINGS

Apart from global settings which are applied to the whole workspace, it is possible to adjust the settings of a separate project. To do this, select Properties from the project menu or use hotkey (Alt + Enter by default), then select pgCodeKeeper.

9.1 Project main settings


Properties for telegram_test

pgCodeKeeper

- [ ] Disable SQL parser for unrelated files opened in SQL Editor
- [x] Use Unix-style newline characters
- [ ] Bind project to database connection: [choose DB]

Timezone for all DB connections: UTC

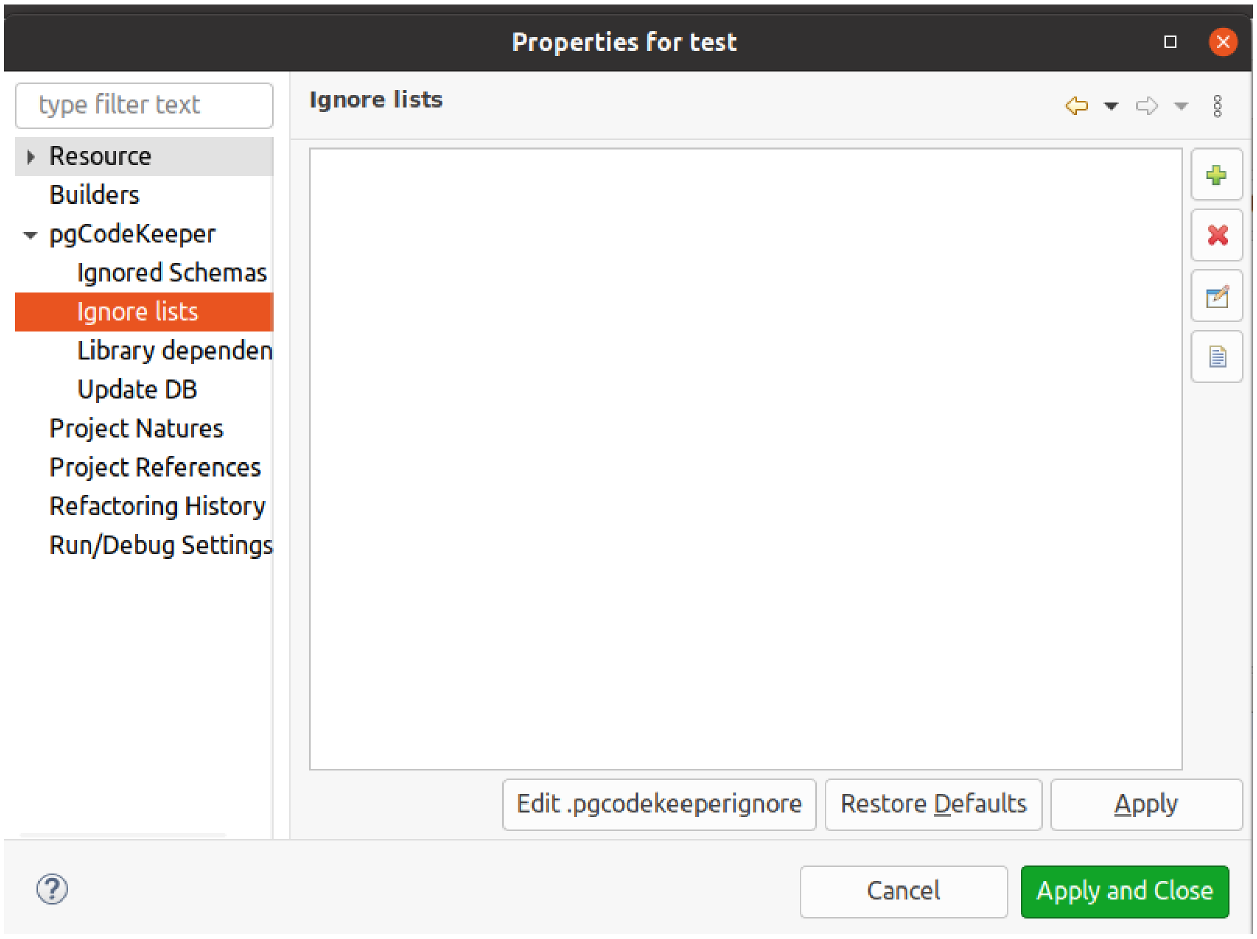
- [ ] Enable project specific settings
- [ ] Ignore differences in table column order
- [x] Enable full dependencies from bodies of functions and procedures (experimental)
- [ ] Ignore privileges and owners of database objects
- [ ] Simple formatting for VIEWs when reading via JDBC (not recommended by PostgreSQL)
- [ ] Format object code automatically
- [x] Use global ignore list

- Disable SQL parser for unrelated files opened in SQL Editor allows disabling pgCodeKeeper builder of the project files, which are beyond standard directories. This disables the search for errors and object references within these files.
- Use Unix-style newline characters allows to use Unix hyphens in lines, for example, in function bodies, object commentaries etc.
- Bind project to database connection - allows binding a project to a specific DB. It will be impossible to change DB to Eclipse tool bar for this project.

- Timezone for all DB connections allows selecting, which time zone to use when connecting to databases. Not included for MS SQL projects.
- Enable project specific settings - allows enabling some of the global settings for the current project.
  - Ignore privileges and owners of database objects - allows disabling the search for differences in the properties of the objects related to DB roles.
  - Ignore differences in table column order - allows to ignore the order of colums when comparing the tables.
  - Enable full dependencies from bodies of functions and procedures (experimental) - allows searching the dependences on functions and procedures within the bodies of other functions and procedures.
  - Simple formatting for VIEWs when reading via JDBC (not recommended by PostgreSQL) - allows using simplified view editing, where all unnecessary brackets are removed from the expression. This format might not be supported in the further PostgeSQL versions.
  - Format object code automatically allows display and save the formatted function code in plpgsql and sql according to the SQL Editor formatting settings. For more info about the code formatting settings, see Formatting.
  - Use global ignore list - allows enabling/disabling the usage of a global list of differences from the settings page Excluded objects.

9.2 Ignore Lists

Settings managing the lists of ignored files for current project. A detailed description of work with the lists see in Ignore List.

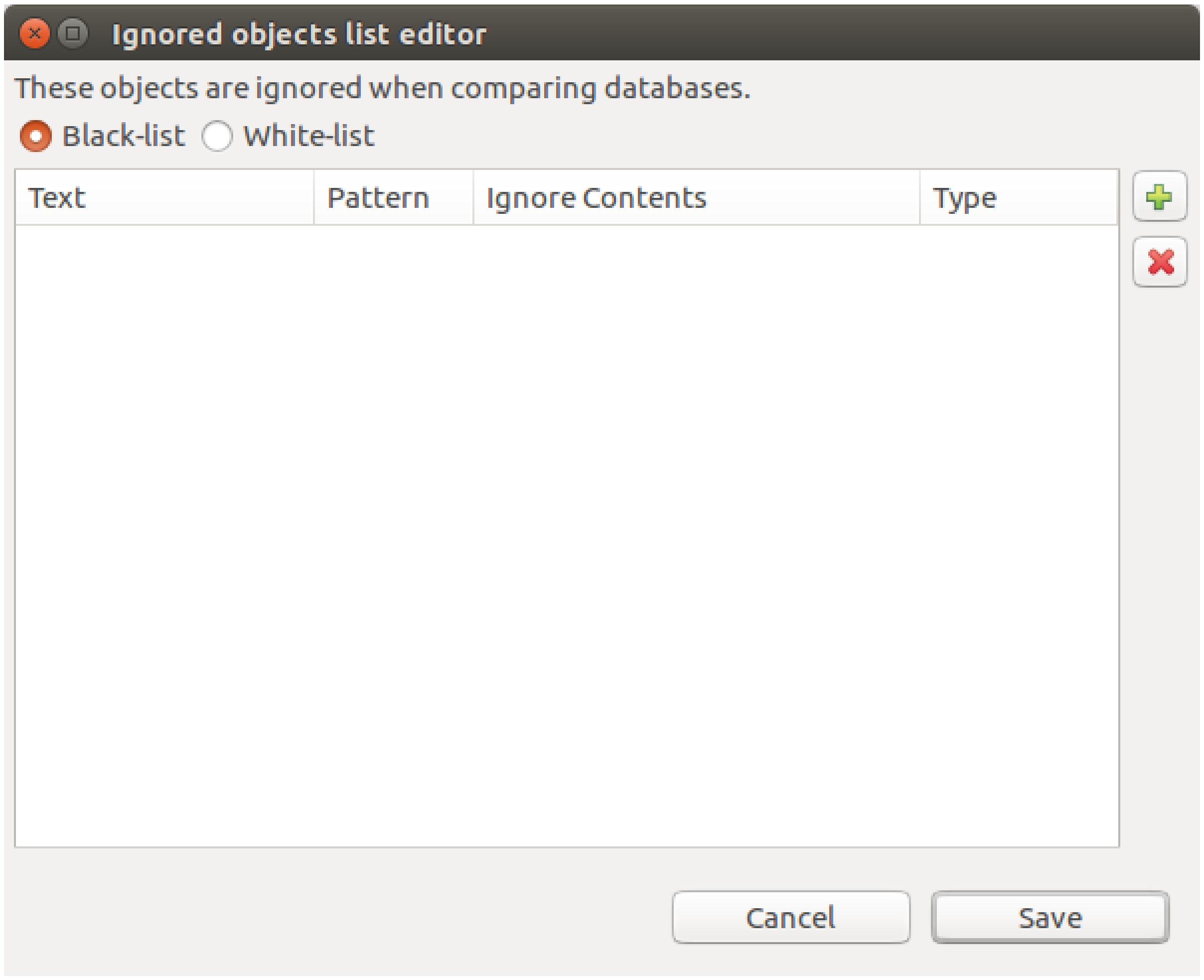


Edit .pgcodekeeperignore allows changing the default ignored file located in the project's root.

To add an existing file, click ➕.

To delete a file, click ✖.

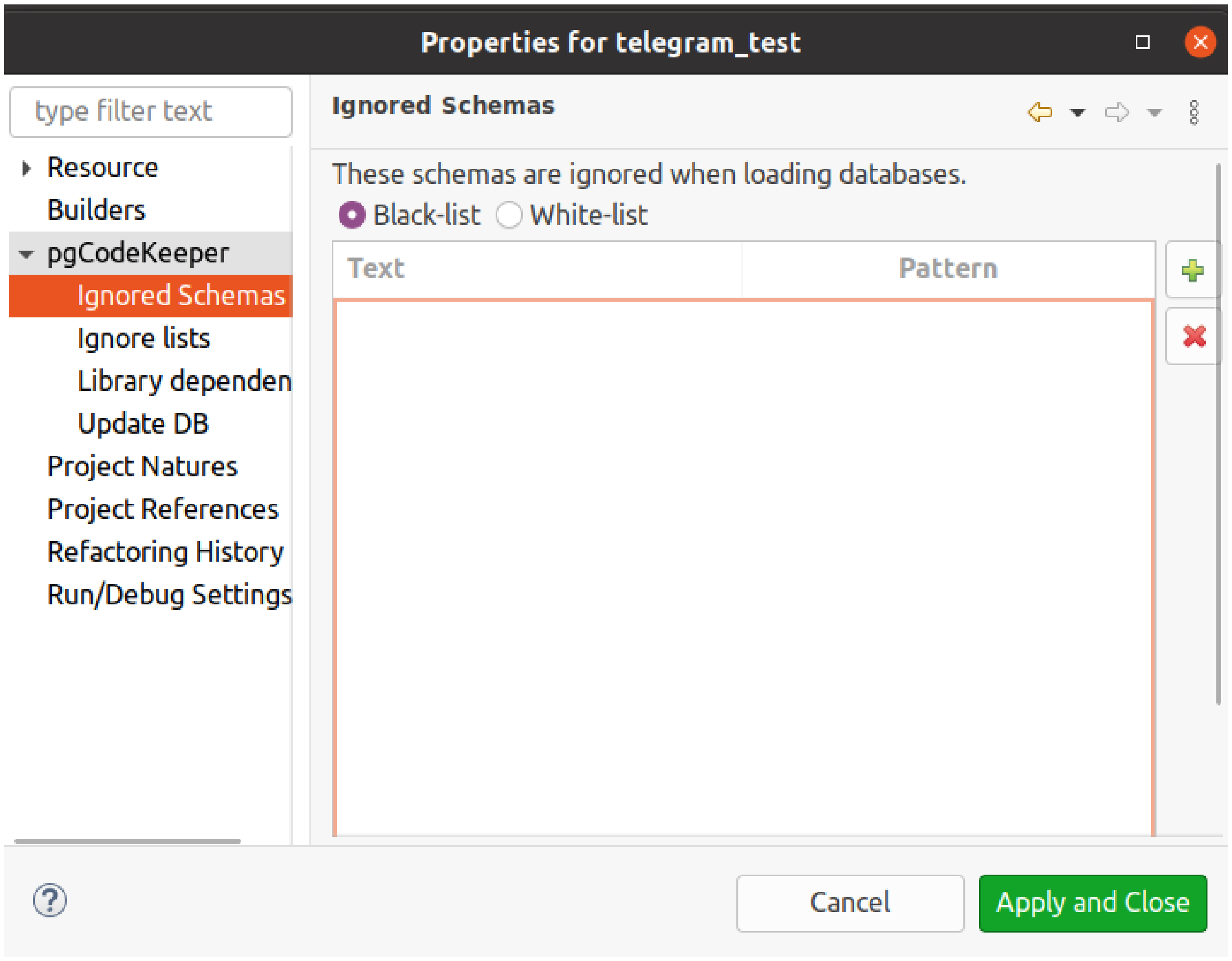
To create a new file, click 📄. The list of rules editor will open, similar to the one on the Excluded objects global settings page.


Ignored objects list editor
These objects are ignored when comparing databases.
Black-list
White-list
Text
Pattern
Ignore Contents
Type
Cancel
Save

To edit a file, click ✎. The editor of list of rules from current file will open.

9.3 Ignored schemas

Settings that regulate the uploading of objects from the database. See the Ignoring schemas when downloading section for more details.



These schemas are ignored when loading databases.

[✓ Black-list] White-list

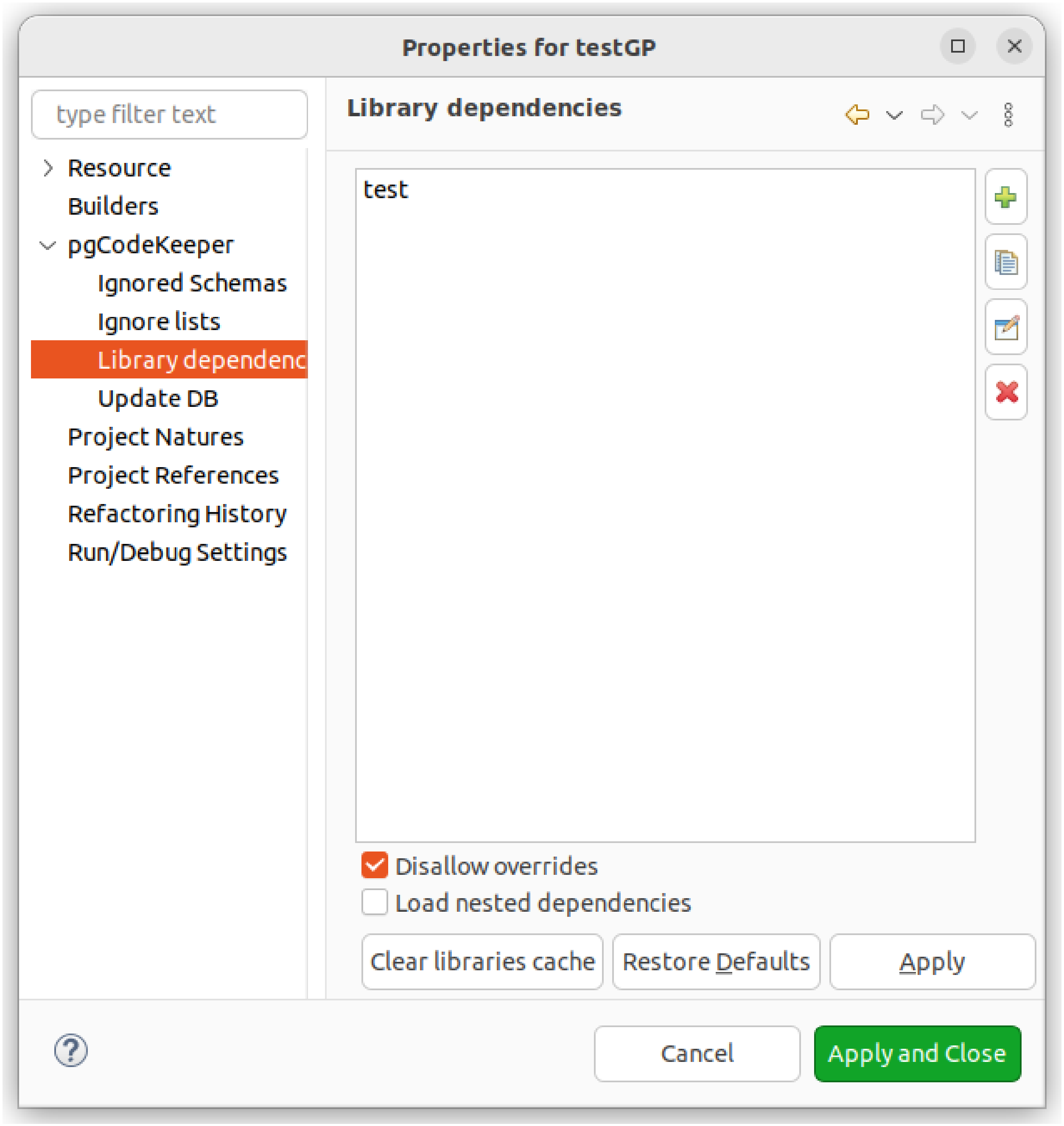
Click ✚ to add a new object. A new object editor will open.

To delete an object, click ✖.

If you need to change an object's name, click the name and enter a new one.

9.4 Library dependences

Settings managing project libraries.



test

- [x] Disallow overrides
- [ ] Load nested dependencies

Libraries are plugged-in sets of object data which are "glued" with project objects during the comparison with the remote database. Directories (a regular directory with files as well as another pgCodeKeeper project), dumps or remote databases can serve as libraries. Besides, you can compress libraries into a zip-archive, upload them to a server and connect them by specifying the archive's URL. To add libraries, use corresponding buttons on the side panel.

The libraries are loaded in the order in which they are indicated on the list (the topmost library will be loaded first). To change the order, use the mouse to drag the required elements.

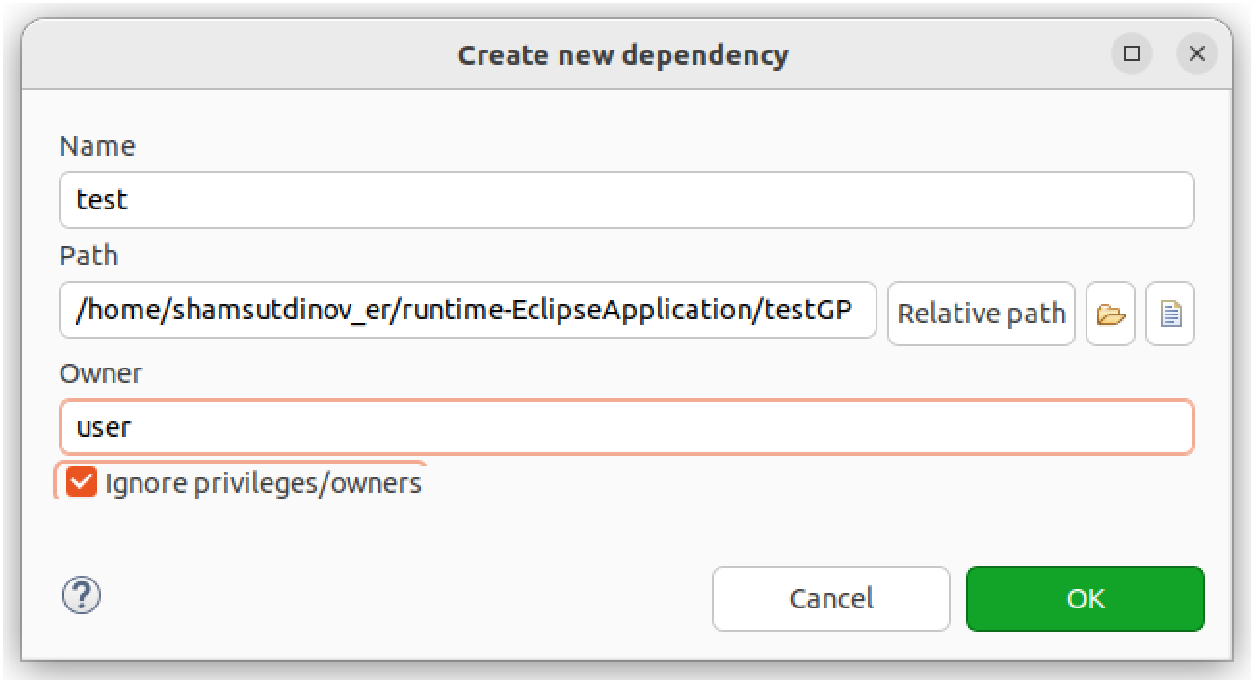
Disallow overrides. There might be a situation when a project and a library (or two libraries) contain objects with the same names. There are two ways of processing such situations. When this settings is on, such conflicts will cause the object comparison to stop. When it's off, it will allow you to ignore the conflicts: the first loaded object will be used. Regardless of the state of the settings, the Object overrides view will be displayed, showing all the conflicts.

Load nested dependencies is a setting that allows loading libraries referred to by other libraries (recursive loading of libraries). The library containing dependencies should be in the form of a project (or project archive) and contain the .dependencies file, which contains the dependencies of this library.

In general, to load the library dependencies, a container of this library should have the loadNested setting in the .dependencies file enabled. For example, in the following dependencies chain: project → lib1 → lib2 → lib3, to load lib2, the setting should be enabled in project, and to load lib3, the setting should be enabled both in project and lib1.

Clear libraries cache - allows to delete the downloaded archives from the cache.

Dialog window of creating/editing/copying the library dependency looks as follows.


Create new dependency

Name: test

Path: /home/shamsutdinov_er/runtime-EclipseApplication/testGP

Owner: user

[✓] Ignore privileges/owners

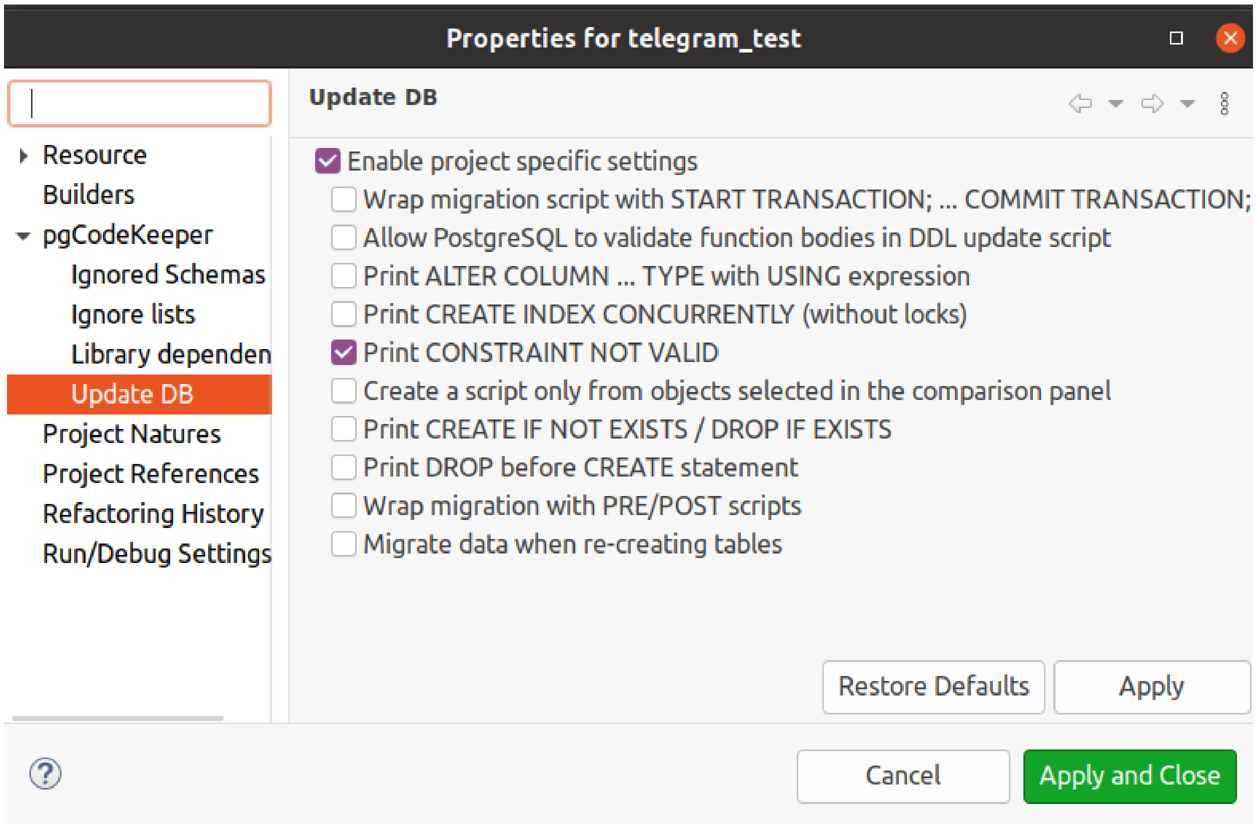
- Name - allows to specify the library name that is going to be displayed (default name is Path).
- Path - allows to specify the library's source (supports relative paths).
- Owner - allows setting a new owner for all the objects in the library.
- Ignore privileges/owners - allows to disable privileges and owners of a library dependency objects.

Important: If a library already has an owner, this information will be displayed regardless of other pgCodeKeeper settings.

Important: If a library doesn't ignore privileges, they will be displayed regardless of other pgCodeKeeper settings.

9.5 DB update

Database update managing settings, which allow enabling some of correlating settings from the DB update page.



Update DB

- [✓] Enable project specific settings
- [ ] Wrap migration script with START TRANSACTION; ... COMMIT TRANSACTION;
- [ ] Allow PostgreSQL to validate function bodies in DDL update script
- [ ] Print ALTER COLUMN ... TYPE with USING expression
- [ ] Print CREATE INDEX CONCURRENTLY (without locks)
- [✓] Print CONSTRAINT NOT VALID
- [ ] Create a script only from objects selected in the comparison panel
- [ ] Print CREATE IF NOT EXISTS / DROP IF EXISTS
- [ ] Print DROP before CREATE statement
- [ ] Wrap migration with PRE/POST scripts
- [ ] Migrate data when re-creating tables

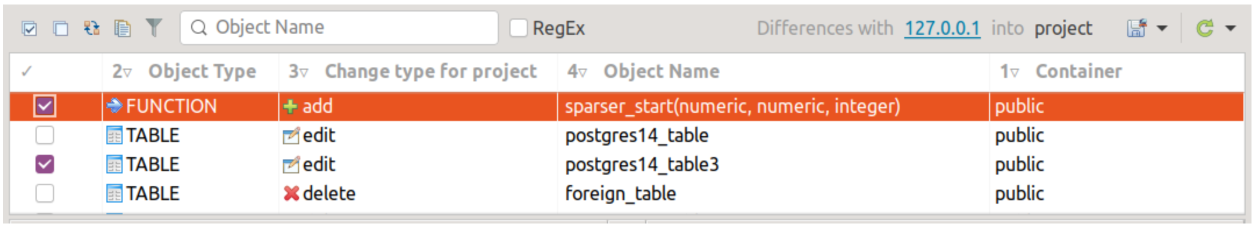
Глава

TEN

PROJECT UPDATE

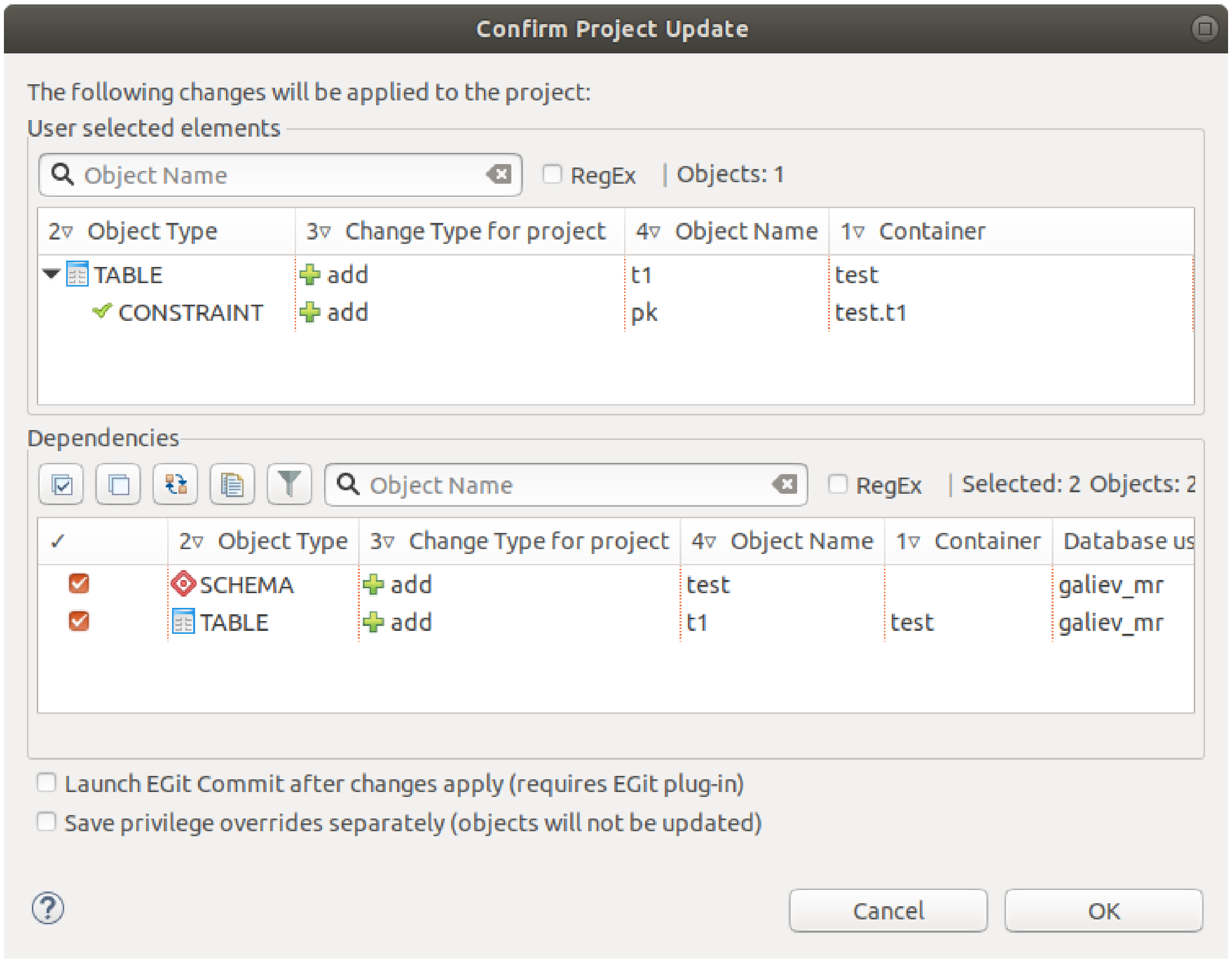
On the tool bar, open the Get Changes drop-down list, and select the database source based on which the pgCodeKeeper project will be updated.

Click . After a short wait, the list of differences with the indication of change type will be displayed on the differences panel.



✓	2▿ Object Type	3▿ Change type for project	4▿ Object Name	1▿ Container
✓	FUNCTION	add	sparser_start(numeric, numeric, integer)	public
	TABLE	edit	postgres14_table	public
✓	TABLE	edit	postgres14_table3	public
	TABLE	delete	foreign_table	public

Open the Apply drop-down menu and select ** Project**. Tick the required changes, click , and the Confirm Project Update dialog window will appear.



The following changes will be applied to the project:

User selected elements

2▿ Object Type	3▿ Change Type for project	4▿ Object Name	1▿ Container
TABLE	add	t1	test
CONSTRAINT	add	pk	test.t1

Dependencies

✓	2▿ Object Type	3▿ Change Type for project	4▿ Object Name	1▿ Container	Database us
✓	SCHEMA	add	test		galiev_mr
✓	TABLE	add	t1	test	galiev_mr

☐ Launch EGit Commit after changes apply (requires EGit plug-in)

☐ Save privilege overrides separately (objects will not be updated)

Check the list of objects selected by the user and the list of dependent objects. .. important:: Be careful! Excluding dependence objects from the update can lead to errors and unexpected results.

If instead of changing the objects themselves you need to save overrides of their properties (for example privileges), select the corresponding option. This can be done only for the objects with the edit change type.

Click OK to confirm the applying of changes to the project. If successful, pgCodeKeeper console will display a corresponding message.

Important: Project file update resets the differences list.

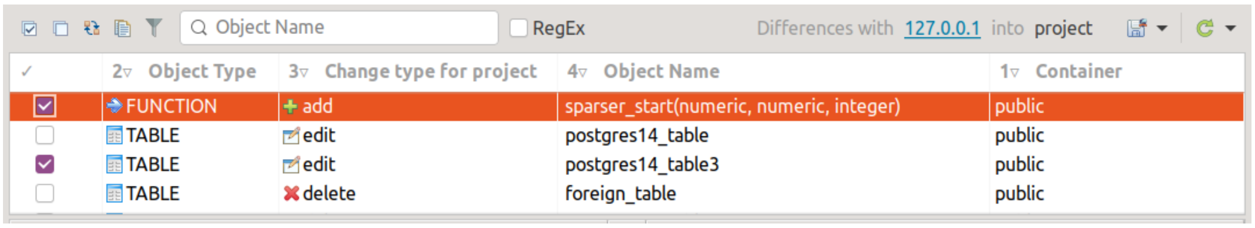
Глава

ELEVEN

UPDATING REMOTE DATABASE

On the project editor tool bar, open the set the Get Changes drop-down menu, or in the drop-down list on the editor tool bar. Next, select the database source based on which a DDL script for applying changes to the remote database will be generated.

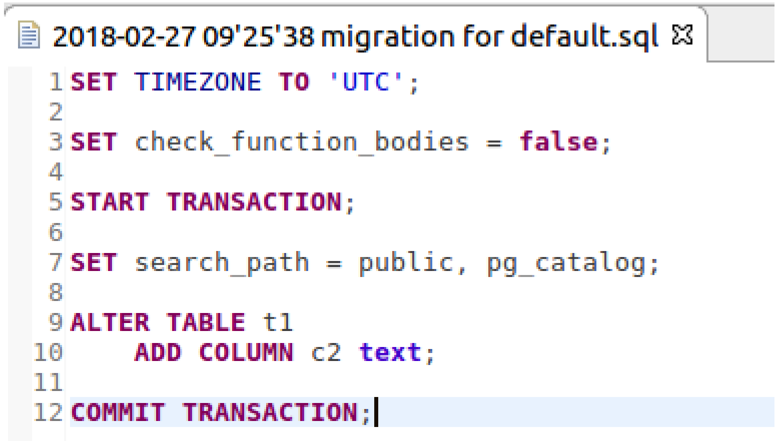
Click . After a short wait, the list of objects with the indication of change type will be displayed on the differences panel.



✓	2▿ Object Type	3▿ Change type for project	4▿ Object Name	1▿ Container
✓	FUNCTION	add	sparser_start(numeric, numeric, integer)	public
	TABLE	edit	postgres14_table	public
✓	TABLE	edit	postgres14_table3	public
	TABLE	delete	foreign_table	public

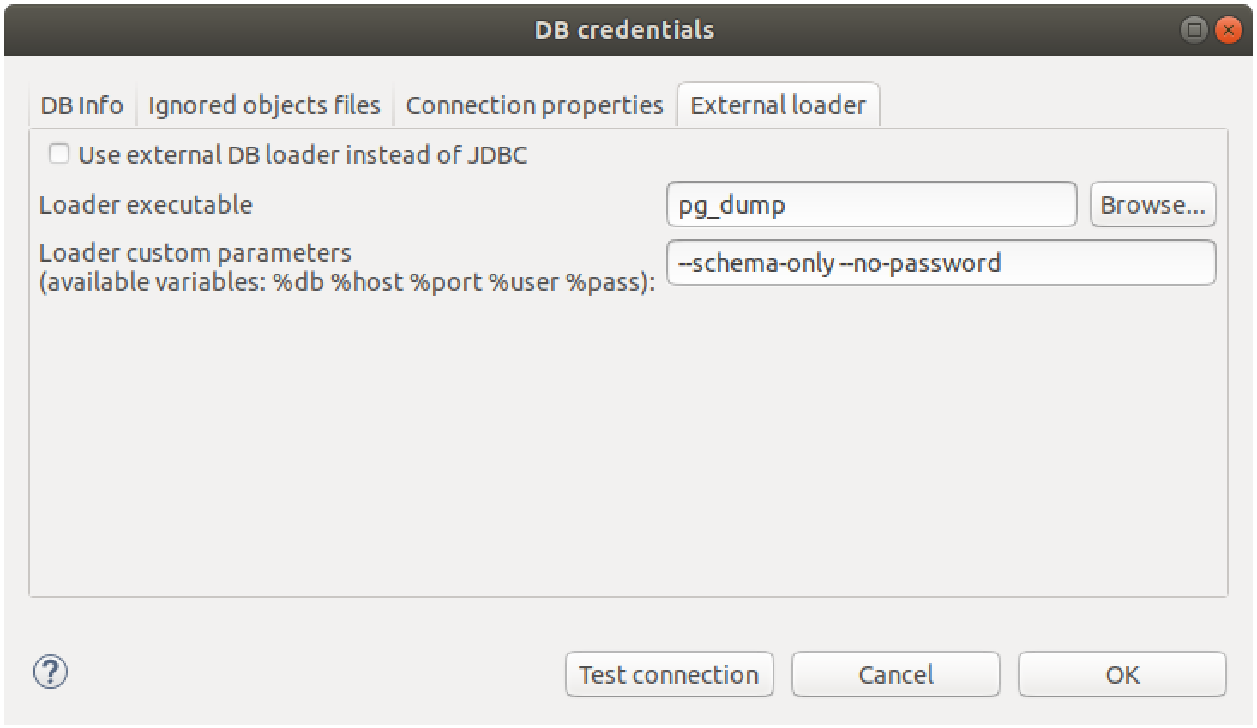
Open the Apply drop-down menu and select DB. You can get the same menu of selecting the direction of changes if you click the Project label. Tick the necessary changes and click .

When script generation is finished, SQL editor will open. Check the generated script, make changes manually if needed.



```
SET TIMEZONE TO 'UTC';

SET check_function_bodies = false;

START TRANSACTION;

SET search_path = public, pg_catalog;

ALTER TABLE t1
    ADD COLUMN c2 text;

COMMIT TRANSACTION;
```

To apply the migration script to the database, specify the method of its execution. Updating via JDBC is used by default. If you want to use another utility or bootup options it is possible to change settings of the current DB. See DB storage for more.

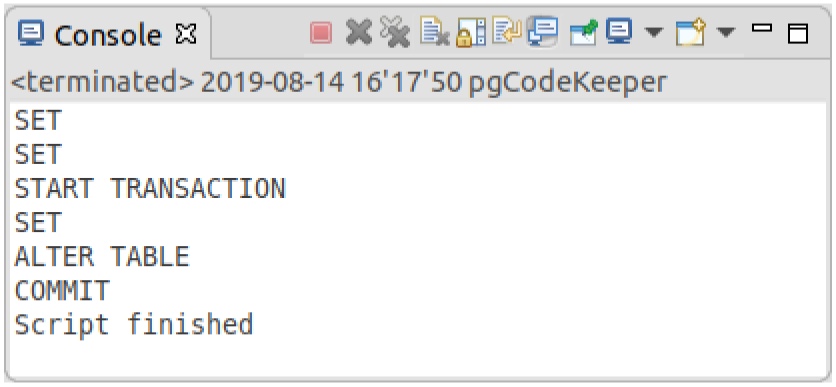


Use external DB loader instead of JDBC

Loader executable: pg_dump

Loader custom parameters (available variables: %db %host %port %user %pass): --schema-only --no-password

After selecting the update method and entering the necessary parameters, click Update DDL or use hotkey (Ctrl + Alt + R by default). Migration script will be applied to the specified database. You can see current progress in the pgCodeKeeper console.



```
<terminated> 2019-08-14 16'17'50 pgCodeKeeper
SET
SET
START TRANSACTION
SET
ALTER TABLE
COMMIT
Script finished
```

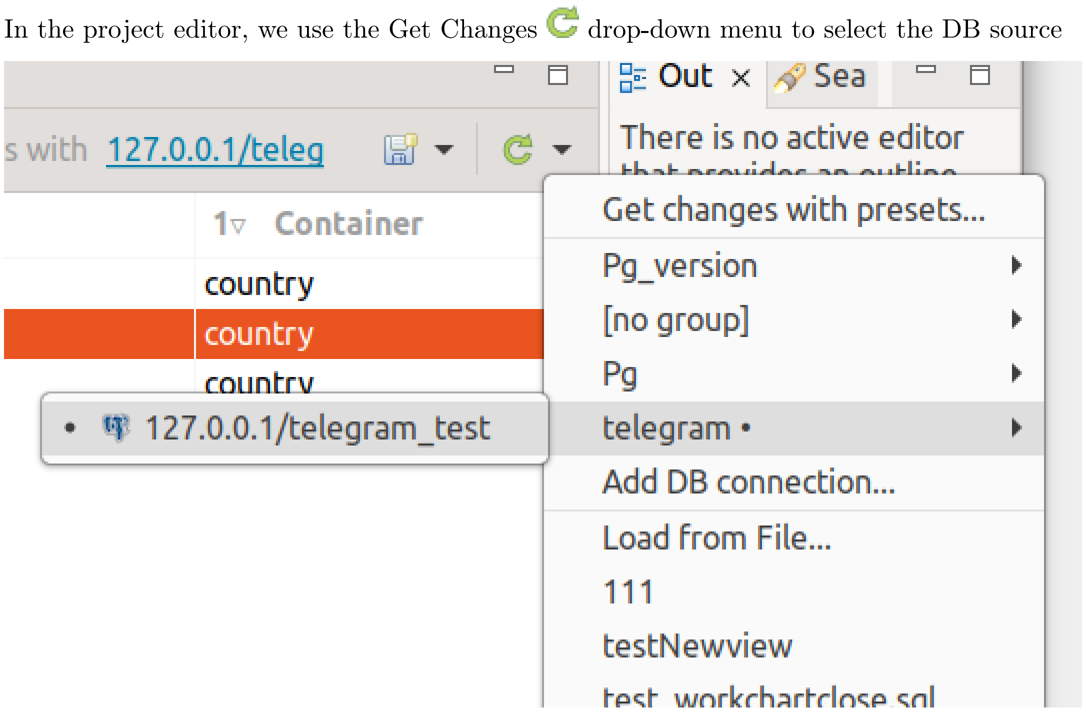
For bigger flexibility, there is an opportunity of migration of a selected fragment of code.

Глава

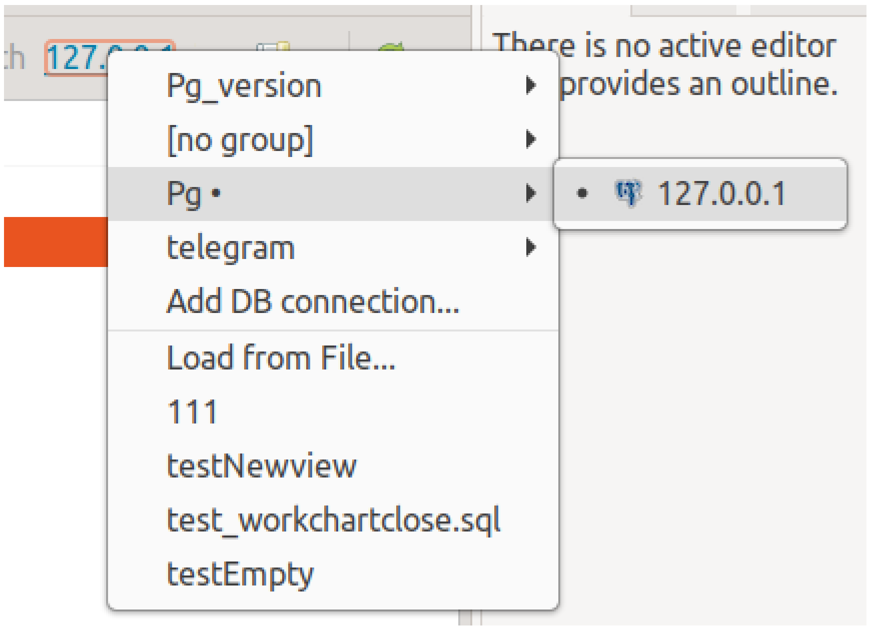
TWELVE

SELECTING OF THE DATABASE SOURCE

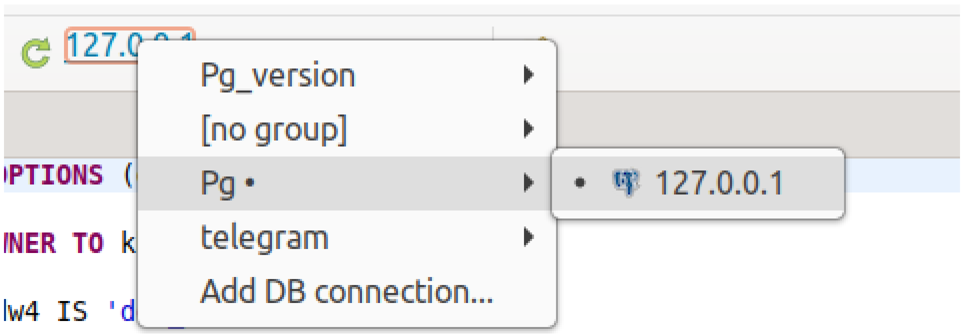
In the project editor, we use the Get Changes drop-down menu to select the DB source



or from the right-click menu of the selected DB.



In the SQL editor, the DB source is selected from the right-click menu on the Eclipse tool bar.



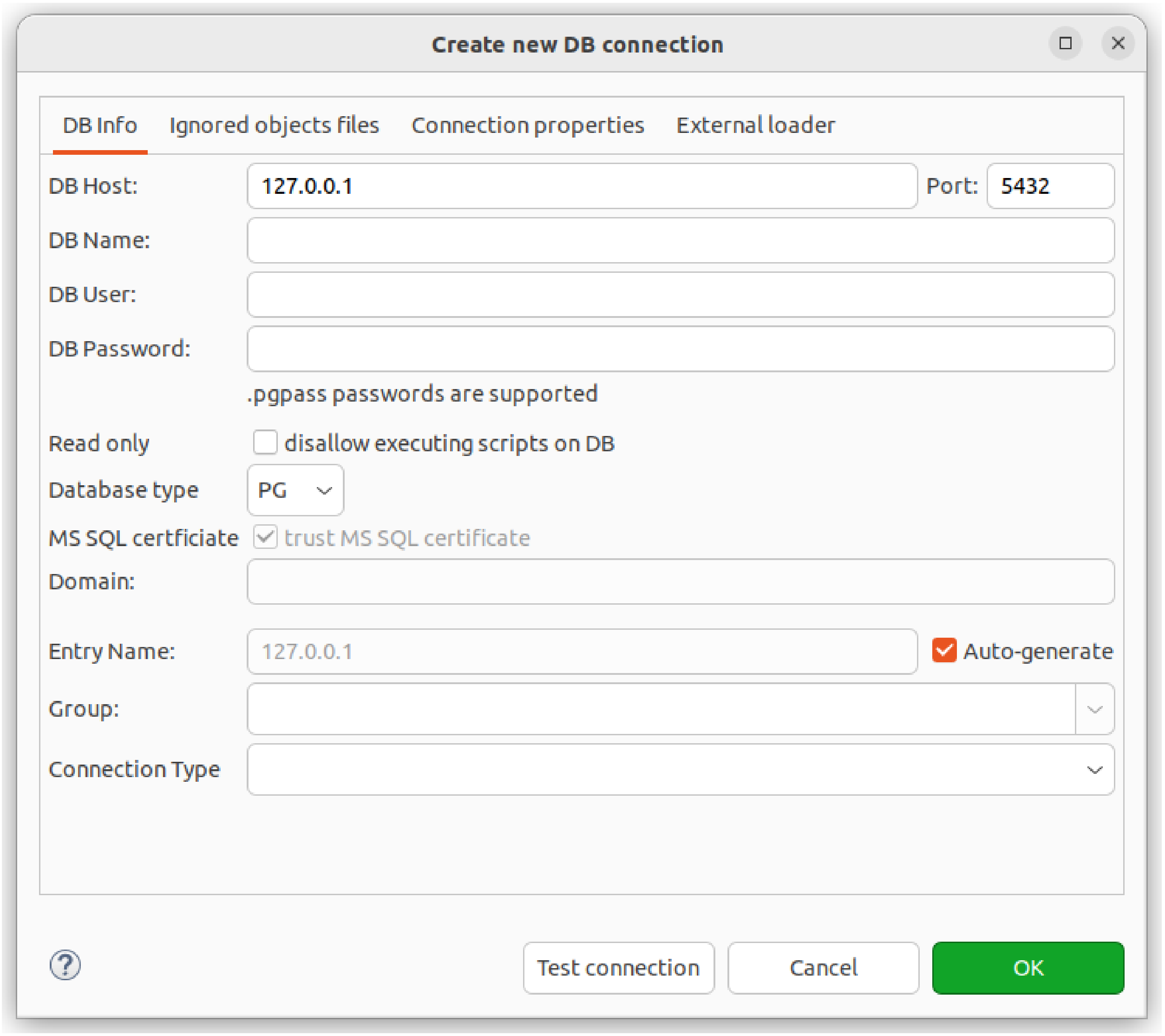
The DB sources lists display the available DB connections merged in groups, as well as the latest used dump files.

You can find the databases list in DB storage, which can be accessed from the Add DB connection item in the Get Changes drop-down list or combo box.

To select a dump file, you should select the item from the Load from file… list. In the appeared dialog window select the required file and click OK. The selected file is added to the database source list.

12.1 Database sources grouping

To merge databases into groups, select DB Store in the pgCodeKeeper global settings and click Edit. In the Group field, specify a group, or select an existing one.



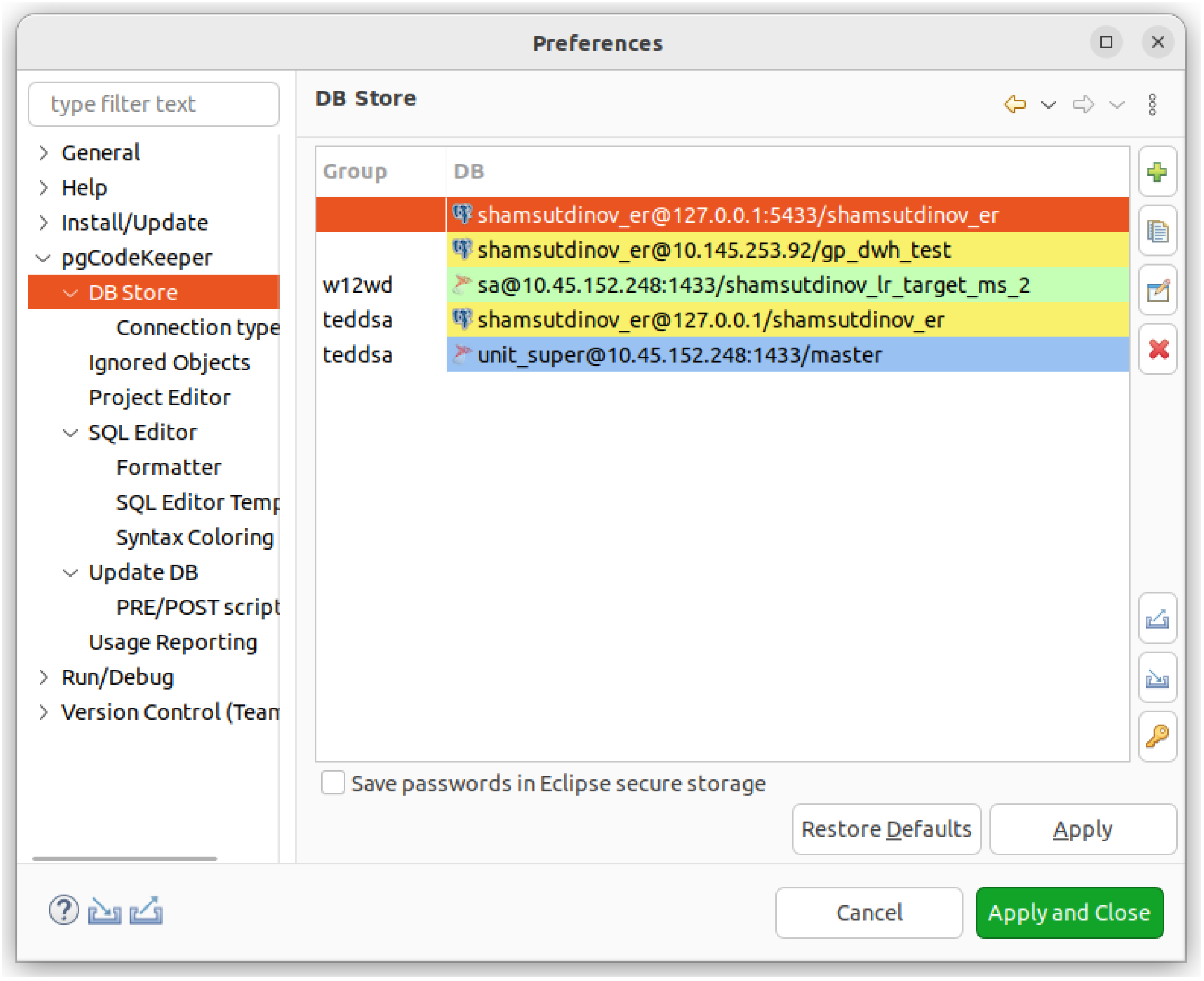
Create new DB connection

DB Info | Ignored objects files | Connection properties | External loader

DB Host:	127.0.0.1 Port: 5432
DB Name:	
DB User:	
DB Password:	
	.pgpass passwords are supported
Read only	☐ disallow executing scripts on DB
Database type	PG
MS SQL certficiate	✓ trust MS SQL certificate
Domain:	
Entry Name:	127.0.0.1 ✓ Auto-generate
Group:	
Connection Type	

12.1.1 Password storage

If Eclipse is unstable, you can disable storing passwords in the Eclipse secure storage. In this case, the DB passwords will be stored as plain text in the Eclipse workspace. You can also use passwords from the .pgpass file. To disable saving passwords in the Eclipse secure storage, uncheck Save password in Eclipse secure storage.



Group	DB
	shamsutdinov_er@127.0.0.1:5433/shamsutdinov_er
	shamsutdinov_er@10.145.253.92/gp_dwh_test
w12wd	sa@10.45.152.248:1433/shamsutdinov_lr_target_ms_2
teddsa	shamsutdinov_er@127.0.0.1/shamsutdinov_er
teddsa	unit_super@10.45.152.248:1433/master

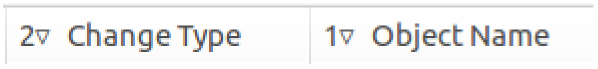
☐ Save passwords in Eclipse secure storage

Глава

THIRTEEN

LIST OF CHANGES SORTING

To sort the objects in the changes list, click the column heading.



By default, multiple sorting is used. When sorting a new column, the sorting of previous selected columns is considered.

To reset the sorting, you can click the column heading while holding Ctrl.

Numbers and arrows in the table heading show the order and direction of sorting.

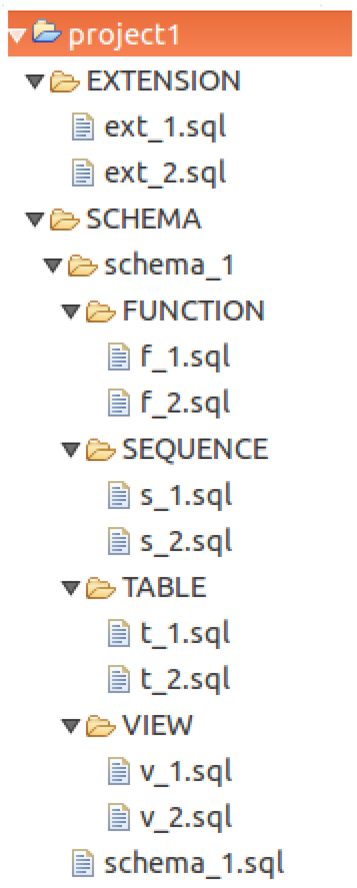
Глава

FOURTEEN

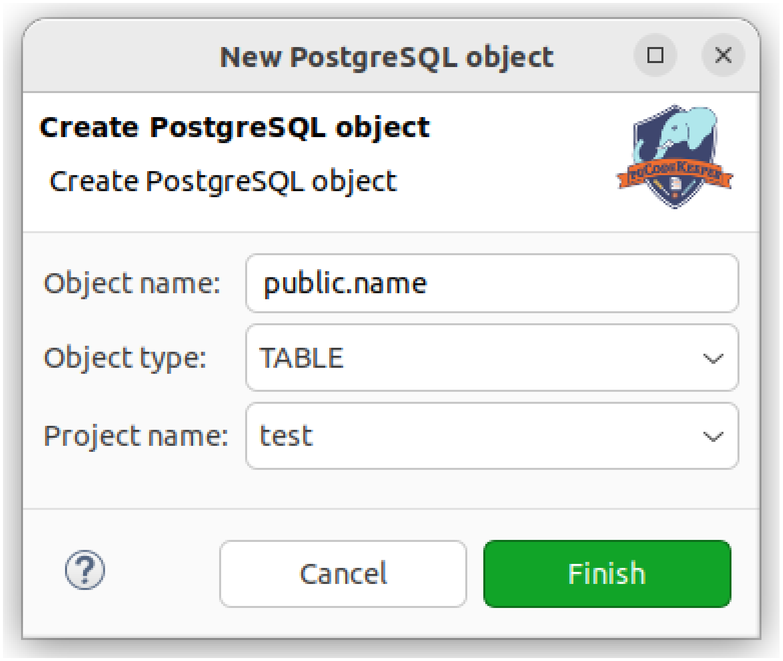
PGCODEKEEPER PROJECT EDITING

Project editing consists of changing the file content, as well as of adding and deleting files and directories which are under a pgCodeKeeper project.

Note: Manual object editing can lead to errors in the program and to incorrect applying of the changes to the DB.



To add objects, we recommend you to use SQL objects creation wizard: File -> New -> SQL object or hotkey (Alt + Insert by default).



Object name: public.name
Object type: TABLE
Project name: test

Select the project and the type of object, enter the name and click Finish.

After that, the SQL editor will appear with a created object template. If the object already exists, it will be opened in the editor.

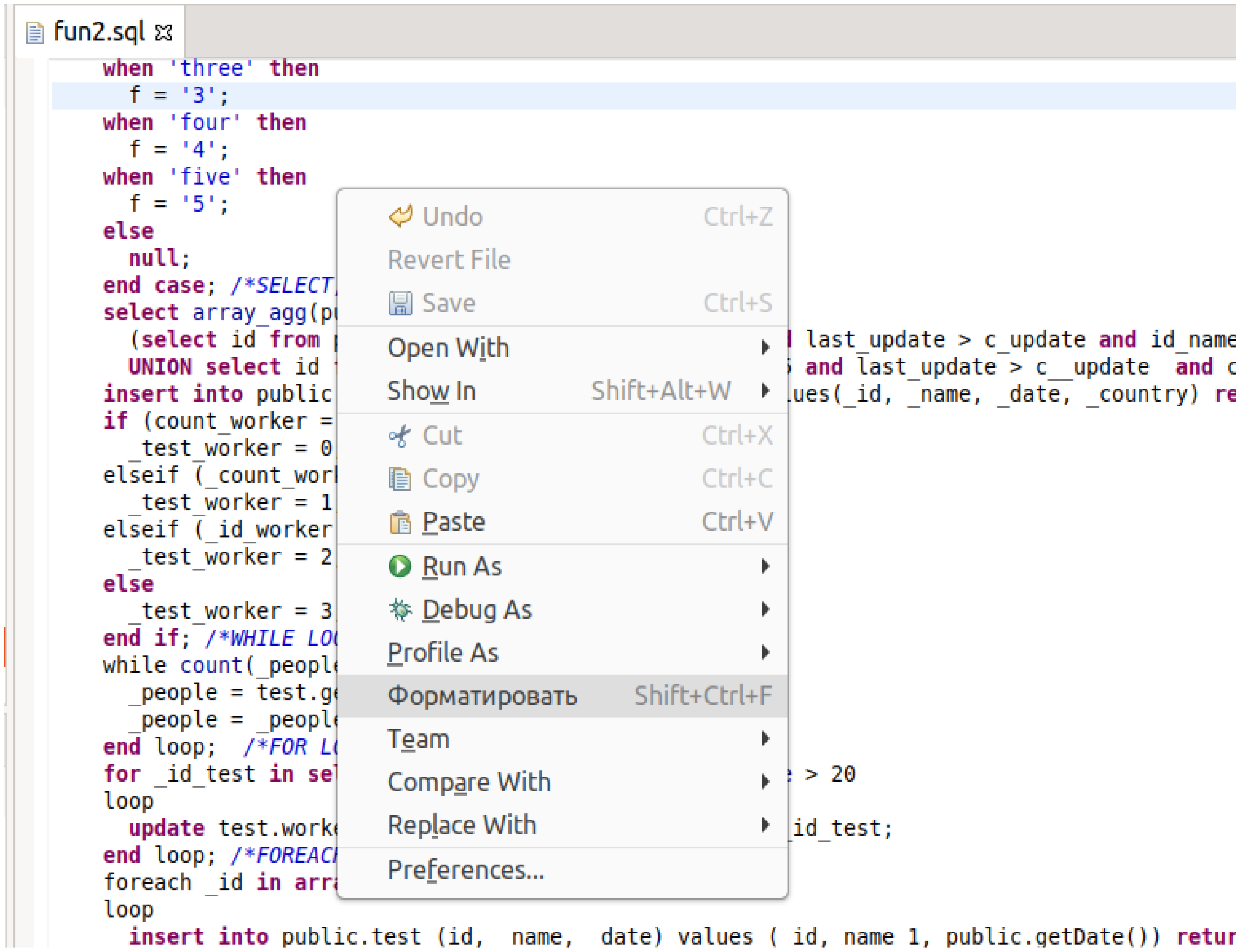
Note: all possible data is filled in automatically. For example: if the wizard was launched with the help of right-click menu of the function file, current project would be selected automatically, type of object would be set to FUNCTION, and the name of object would be partially filled by the name of schema in which the function is contained.

To edit an existing object, it is sufficient to change the content of its file in the project.

FIFTEEN

FORMATTING

Formatting is automatically applied to the code after you press Shift+Ctrl+F or select Format in the right-click menu. Formatting settings are stored in the global settings in Formatting.



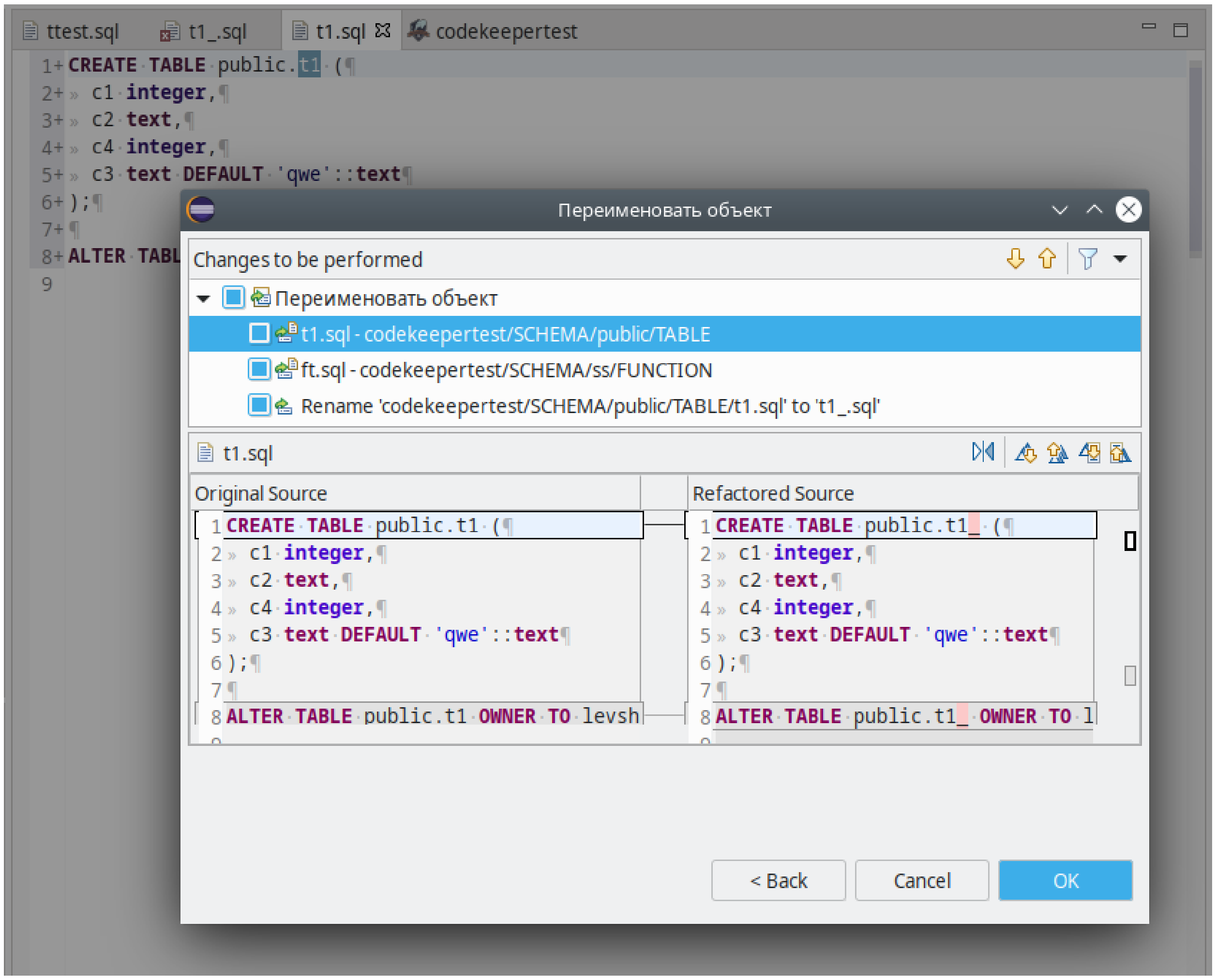
```
    when 'three' then
      f = '3';
    when 'four' then
      f = '4';
    when 'five' then
      f = '5';
    else
      null;
    end case; /*SELECT
    select array_agg(pu
      (select id from                last_update > c_update and id_name
      UNION select id                and last_update > c__update  and c
    insert into public               ues(_id, _name, _date, _country) re
    if (count_worker =
       _test_worker = 0
    elseif (_count_wor
       _test_worker = 1
    elseif (_id_worker
       _test_worker = 2
    else
       _test_worker = 3
    end if; /*WHILE LO
    while count(_people
      _people = test.ge
      _people = _people
    end loop;  /*FOR L
    for _id_test in se               > 20
    loop
      update test.worke              id_test;
    end loop; /*FOREACH
    foreach _id in arra
    loop
      insert into public.test (id,  name,  date) values ( id, name 1, public.getDate()) retur
```

Глава

SIXTEEN

REFACTORING

It is possible to refactor the SQL code in pgCodeKeeper. Objects within the SQL code can be renamed along with the links to them. To rename an object, select Rename in the SQL editor's right-click menu or press Alt+Shift+R.



```
CREATE TABLE public.t1 (
	c1 integer,
	c2 text,
	c4 integer,
	c3 text DEFAULT 'qwe'::text
);

ALTER TABL
```

Original Source:

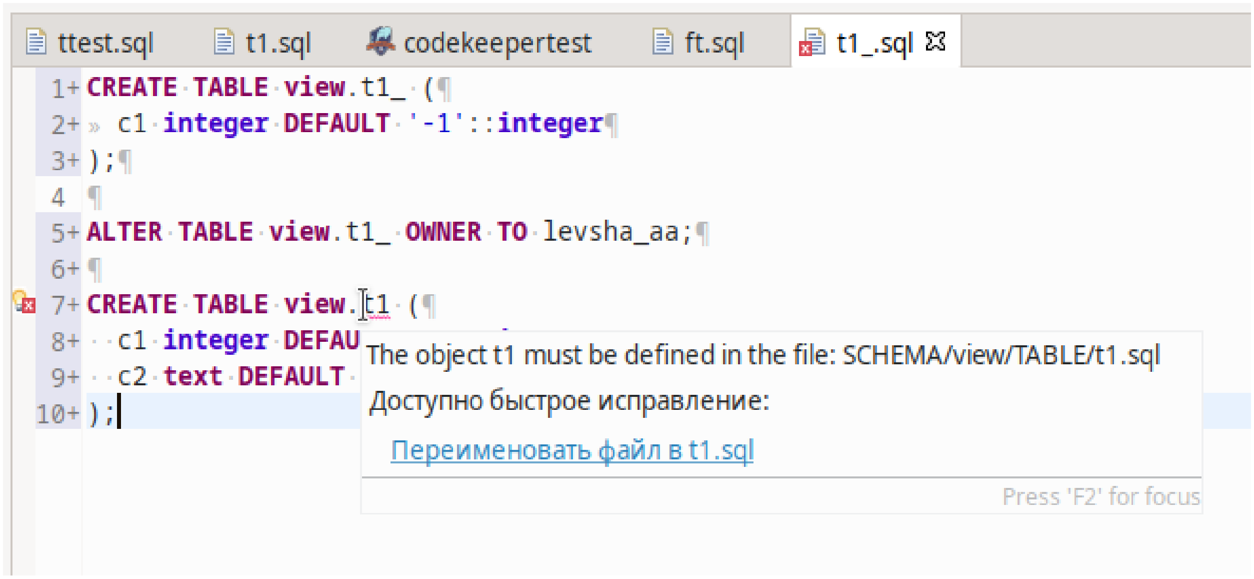
```
CREATE TABLE public.t1 (
	c1 integer,
	c2 text,
	c4 integer,
	c3 text DEFAULT 'qwe'::text
);

ALTER TABLE public.t1 OWNER TO levsh
```

Refactored Source:

```
CREATE TABLE public.t1_ (
	c1 integer,
	c2 text,
	c4 integer,
	c3 text DEFAULT 'qwe'::text
);

ALTER TABLE public.t1_ OWNER TO l
```

16.1 Quick fixes

pgCodeKeeper also allows you to quickly fix errors related to an invalid location of objects. To do that, hover over the error (in the file name) and select Rename file in the pop-up message or the SQL editor's Quick Fix right-click menu. The object's file will be moved to its expected location within the project.



```
CREATE TABLE view.t1_ (
	c1 integer DEFAULT '-1'::integer
);

ALTER TABLE view.t1_ OWNER TO levsha_aa;

CREATE TABLE view.t1 (
  c1 integer DEFAU
  c2 text DEFAULT
);
```

The object t1 must be defined in the file: SCHEMA/view/TABLE/t1.sql
Доступно быстрое исправление:
Переименовать файл в t1.sql
Press 'F2' for focus

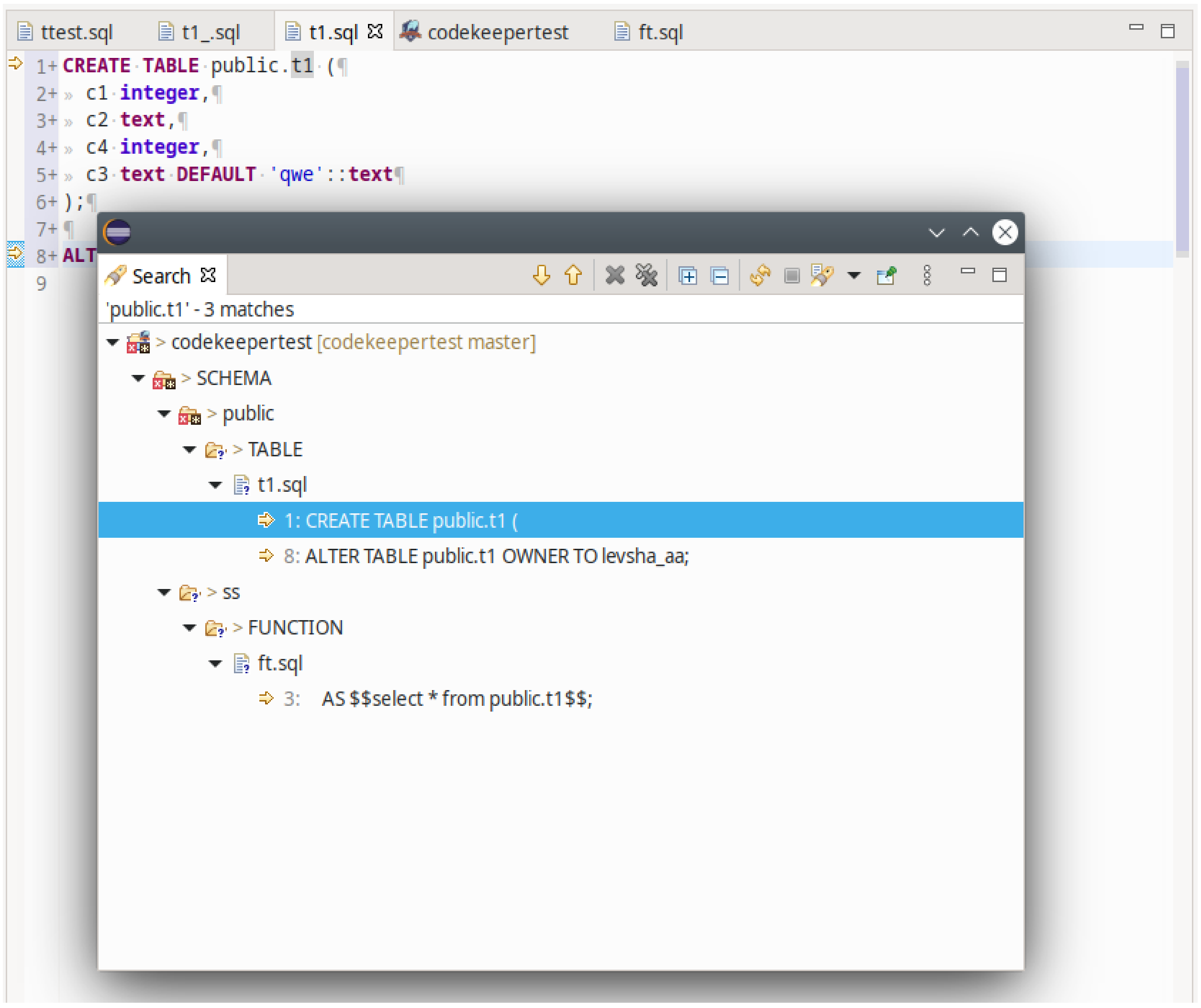
Глава

SEVENTEEN

OBJECT SEARCH

17.1 Search for links to objects

To find all the links to a required object, you should select this object and press Ctrl+Shift+G, or click Find references in the SQL editor's right-click menu



```
CREATE TABLE public.t1 (
	c1 integer,
	c2 text,
	c4 integer,
	c3 text DEFAULT 'qwe'::text
);

ALT
```

'public.t1' - 3 matches

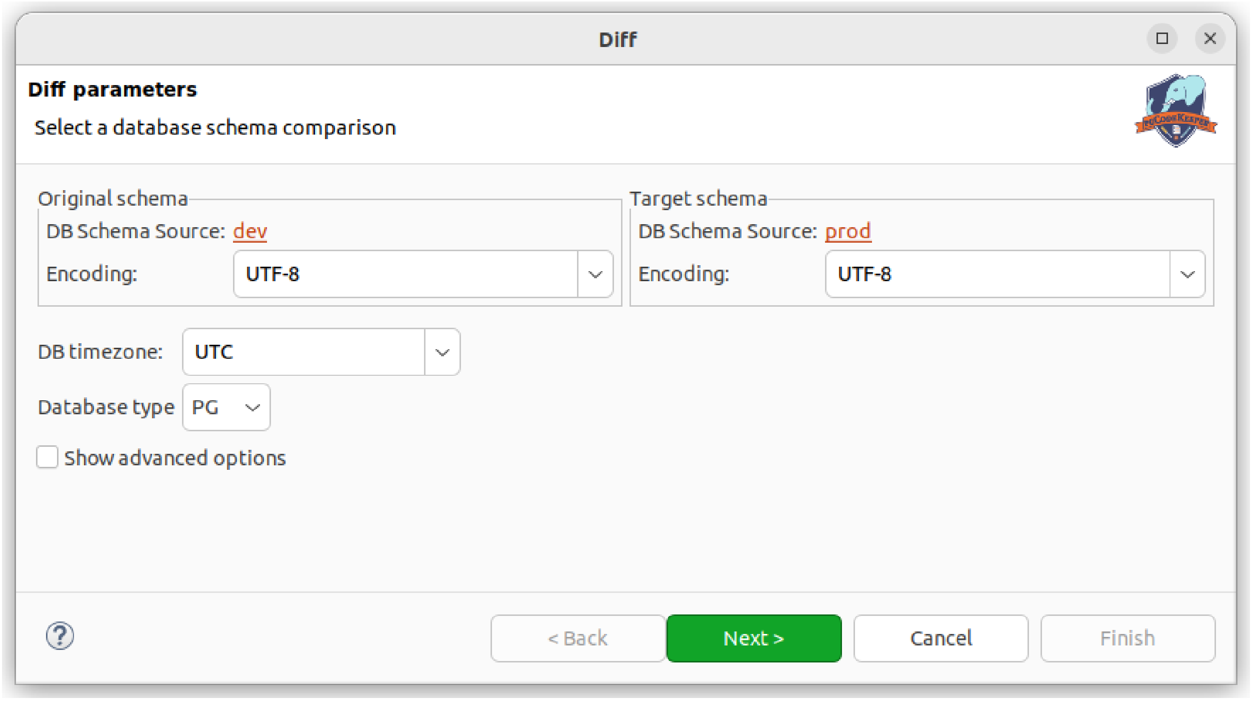
```
> codekeepertest [codekeepertest master]
  > SCHEMA
    > public
      > TABLE
        t1.sql
          1: CREATE TABLE public.t1 (
          8: ALTER TABLE public.t1 OWNER TO levsha_aa;
    > ss
      > FUNCTION
        ft.sql
          3:    AS $$select * from public.t1$$;
```

Глава

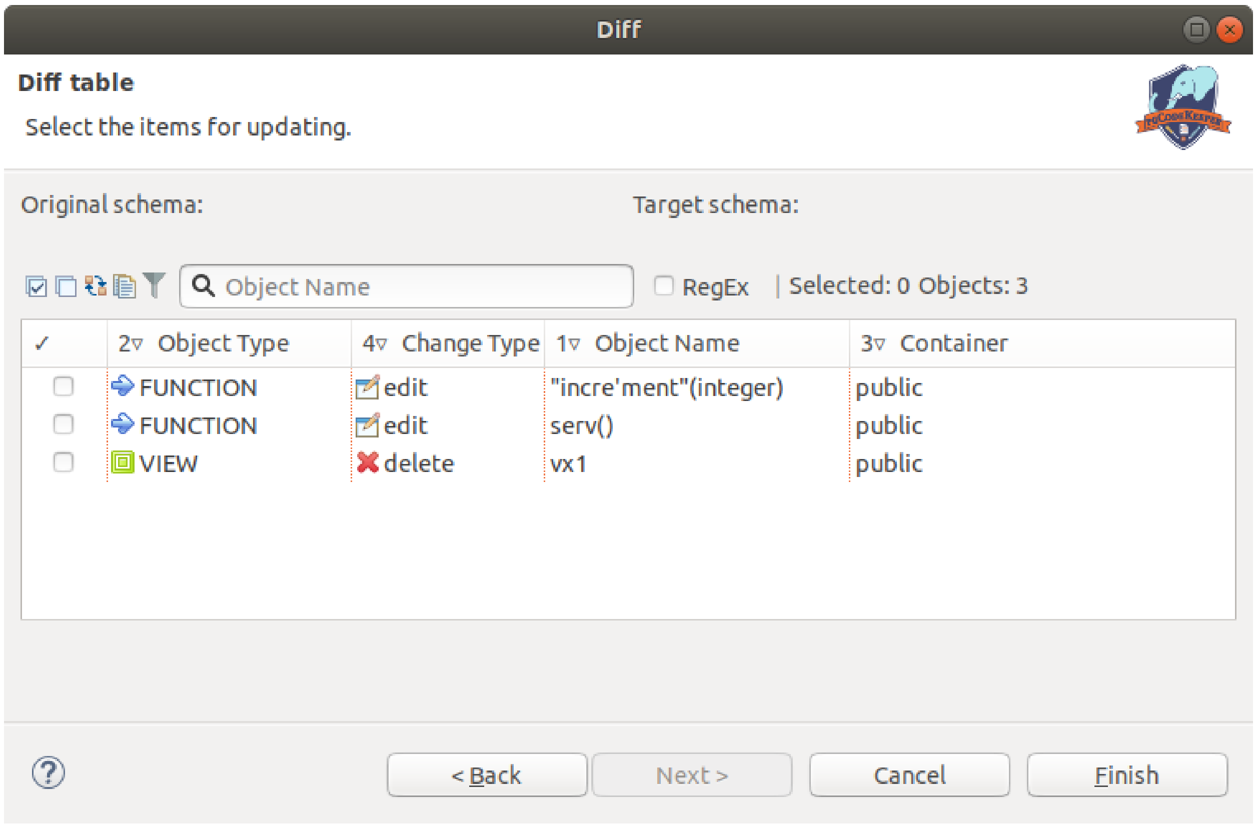
EIGHTEEN

DB SCHEMAS COMPARISON

- Start the database diff wizard: pgCodeKeeper - Diff Wizard or right-click for the pgCodeKeeper project menu and select pgCodeKeeper - Diff. . . .



- Select the source of the original database schema which you want to update and the target database schema, basing on which you are going to update the original one.
- Select source encodings and DB timezone.
- Select the Database type.
- If you need to override the global settings, you can use the Show advanced options option.
- Click Next.



✓	2▿ Object Type	4▿ Change Type	1▿ Object Name	3▿ Container
☐	FUNCTION	edit	"incre'ment"(integer)	public
☐	FUNCTION	edit	serv()	public
☐	VIEW	delete	vx1	public

In the comparison table, select the differences that should be applied. Note: tool bar works similarly to the pgCodeKeeper project editor tool bar.

After clicking Finish, SQL editor with migration script, generated for the selected changes, will open.

Глава

NINETEEN

TEST DATA GENERATION

There is a way of test data generation for the table in the form of the Insert construction:

Open the new object creation wizard: File -> New -> Others, select the pgCodeKeeper -> Generate test data category and click Next.

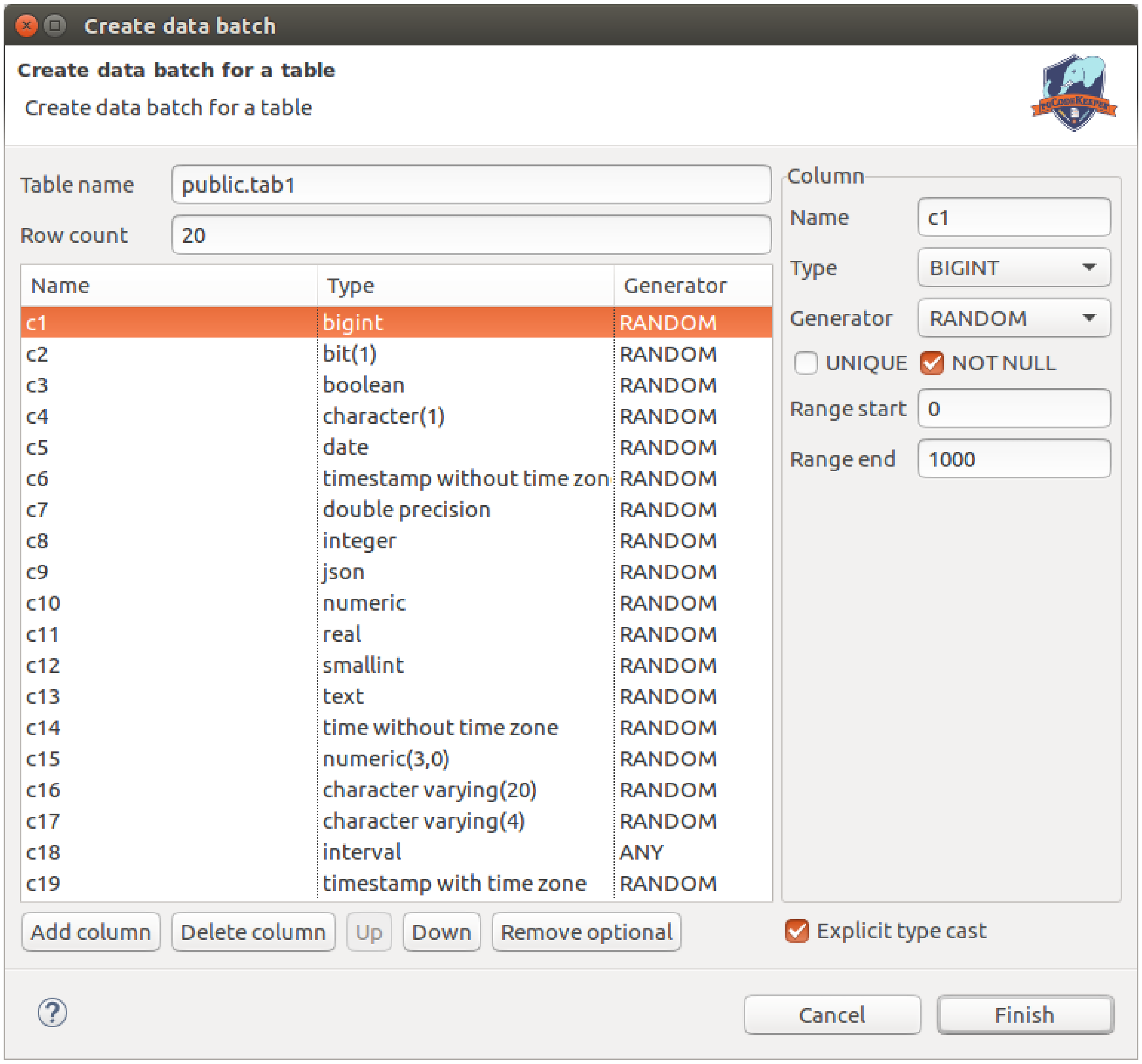


Table name: public.tab1

Row count: 20

Name	Type	Generator
c1	bigint	RANDOM
c2	bit(1)	RANDOM
c3	boolean	RANDOM
c4	character(1)	RANDOM
c5	date	RANDOM
c6	timestamp without time zon	RANDOM
c7	double precision	RANDOM
c8	integer	RANDOM
c9	json	RANDOM
c10	numeric	RANDOM
c11	real	RANDOM
c12	smallint	RANDOM
c13	text	RANDOM
c14	time without time zone	RANDOM
c15	numeric(3,0)	RANDOM
c16	character varying(20)	RANDOM
c17	character varying(4)	RANDOM
c18	interval	ANY
c19	timestamp with time zone	RANDOM

Column

Name: c1

Type: BIGINT

Generator: RANDOM

☐ UNIQUE ☑ NOT NULL

Range start: 0

Range end: 1000

☑ Explicit type cast

For each column, depending on type and generator, the following settings are available:

- Column name - the name of current column.
- Column type - data type of the column. For unsupported types there is the OTHER type.
- Generator - data generation algorithm. Currently there are 4 of them:

  RANDOM
  : A random value from the range.

  INCREMENT
  : Consistent increasing of the inital value by the increment value.

  CONSTANT
  : A constant value. Text values will be enclosed in single quotes.

  ANY
  : Any value (including function calls). The value will not be formatted.

- Range start - initial value
- Range end - final value
- Increment step - increment step value
- Undefined value - unidentified value which will be written without formatting.

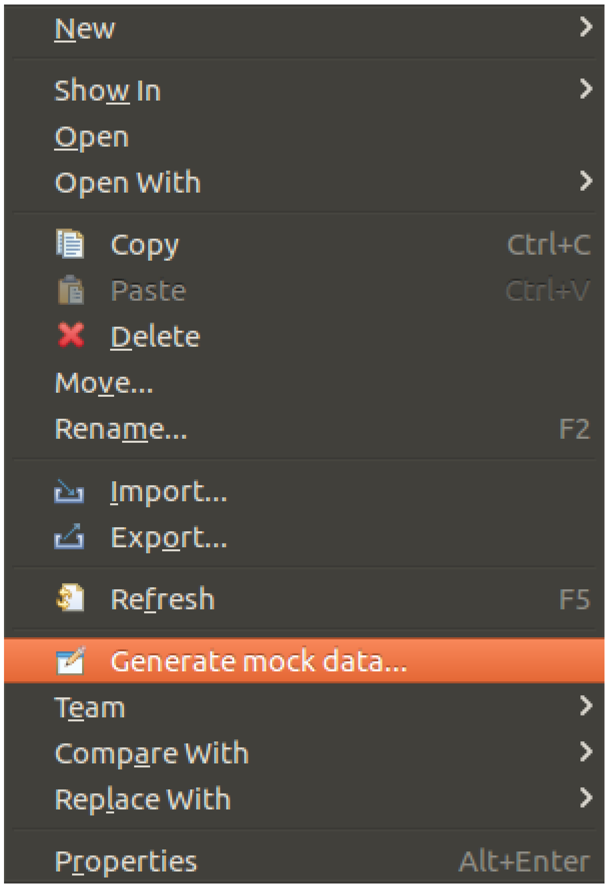
To manage the column list, the following buttons are available:

- Add column adds an INTEGER type column with the RANDOM generator.
- Delete column deletes the selected columns.
- Up moves the selected column up in the list.
- Down moves the selected column down in the list.
- Delete optional deletes the columns without the NOT NULL property.

The Typcasting setting adds typecasting for each value (for example, 7::integer).

After clicking Finish you'll see the SQL editor with a ready Insert query.

Note: This wizard may be launched from the table's right-click menu as well. In this case, all the data is filled in automatically.


New
Show In
Open
Open With
Copy Ctrl+C
Paste Ctrl+V
Delete
Move...
Rename... F2
Import...
Export...
Refresh F5
Generate mock data...
Team
Compare With
Replace With
Properties Alt+Enter

Глава

TWENTY

IGNORE LIST

Ignore List is a list of objects that should or should not be displayed when comparing databases.

Attention: Objects filtered with the help of these lists participate in the comparison and can end up in the resulting migration script of the dependance.

There are two types of lists of ignored objects:

black list
: allows displaying all the objects except the ones indicated in the rules of the list;

white list
: in opposition to black list, forbids displaying all the objects except the ones indicated in the rules of the list.

In each project, it is possible to create a ignore list, applying only to current project. To do this, create the .pgcodekeeperignore file in the project root and fill it with the rules according to the list syntax.

Attention: The .pgcodekeeperignore file format is case-sensitive. You should use UTF-8.

In the following, in the List syntax section, we will consider the rules of forming the .pgcodekeeperignore file.

Note: In addition to the .pgcodekeeperignore file, in the graphical version of pgCodeKeeper, there is a general list of ignored objects. It affects all the projects located in the project directory. You can fill it on the global settings page in the Excluded objects section.

20.1 List syntax

.pgcodekeeperignore consists of two parts:

title
: is obligatory, occupies the first line and determines type of the list;

rules
: lines of rules which forbid or allow (depending on the type of list) displaying some object.

Black list allows to display only the objects which are not specified by rules. It is constructed so that the heading of the list allows displaying all the objects, and the rules following it exclude the objects from displaying.

```
# комментарий
SHOW ALL
HIDE flag [, ...] objectName [ db=dbName ] [ type=objectType [, ...] ] # комментарий
[ ... ]
```

Description of black list components

.pgcodekeeperignore file line	Description
SHOW ALL	[title] allows displaying all the objects
HIDE flag [, ...] objectName [db=dbName][type=objectType[, ...]]	[rule] excludes from displaying an object

White list, on the contrary, allows to display only the objects which are specified by rules. It is constructed so that the heading of the list forbids displaying all the objects, and the rules following it allow to display the objects.

```
# комментарий
HIDE ALL
SHOW flag [, ...] objectName [ db=dbName ] [ type=objectType [, ...] ] # комментарий
[ ... ]
```

Description of white list components

.pgcodekeeperignore file line	Description
HIDE ALL	[title] forbids displaying all the objects
SHOW flag [, ...] objectName [db=dbName] [type=objectType [, ...]]	[rule] allows to display an object

Where:

objectName, dbName, objectType

identifiers for object search identical to the PostgreSQL identifiers to use special characters (for example, in regular expressions), the identifier should be put in quotation marks

see the identifier rule below for more info

objectName	name of the target object (is an obligatory part of the rule)
dbName	name of the database where the search for the object will take place (optional)
objectType	type of the target object (optional)

Important: objectType may be equal to one of the following values: CAST, USER, ROLE, ASSEMBLY, SCHEMA, EXTENSION, TYPE, DOMAIN, SEQUENCE, OPERATOR, FTS_PARSER, FTS_TEMPLATE, FTS_DICTIONARY, FTS_CONFIGURATION, TABLE, FUNCTION, PROCEDURE, AGGREGATE, CONSTRAINT, VIEW, INDEX, TRIGGER, RULE, POLICY.

identifier

identifier string without quotes, consisting of latin characters, numerals and underlining;

can't start with a numeral;

identifier string can't be a key word (reserved language directive)

or

line limited by double quotes `"` or apostrophes `'` (separation symbols in the beginning and the end of the line should match);

separation symbols contained in the line, are escaped by repeating the symbol (for example: `"1""2'3"` или `'1"2''3'`)

Attention: "objectType" is case-sensitive.

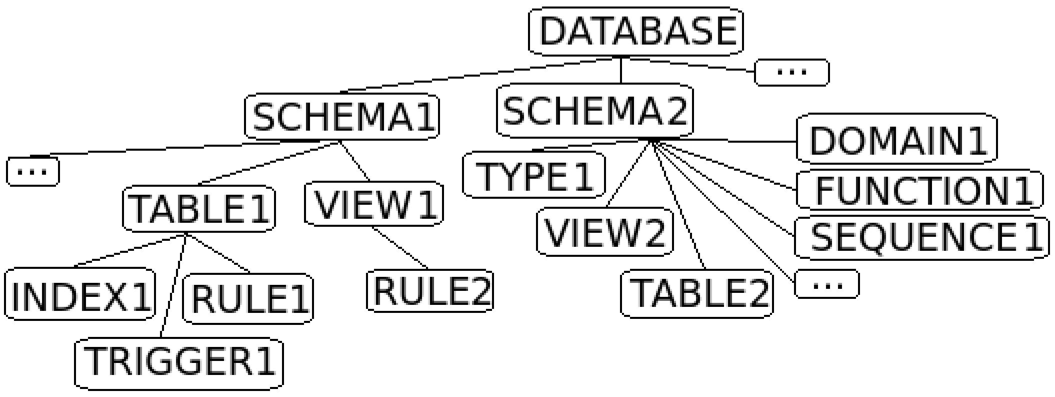
flag { NONE | REGEX | CONTENT | QUALIFIED }

flag of ways of searching the object

REGEX	treats objectName as a regular expression[1]
CONTENT	applies the rule for the matched object and all of its content
QUALIFIED	compares objectName to the qualified name of the object
NONE	complete match without the regular expression and content

Content is the object content, according to the pgCodeKeeper project hierarchy:

[1] REGEX flag allows the rule to search for partial matches with the regular expression.



For example, “INDEX1”, “TRIGGER1” and “RULE1” will be the content of “TABLE1”.

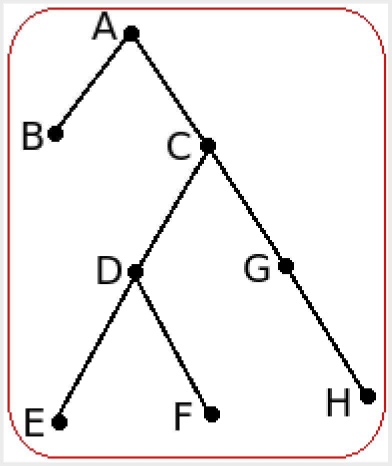
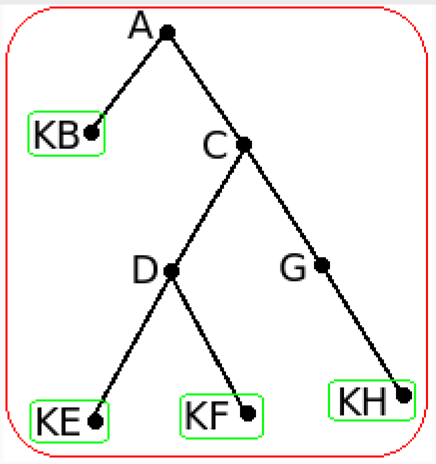
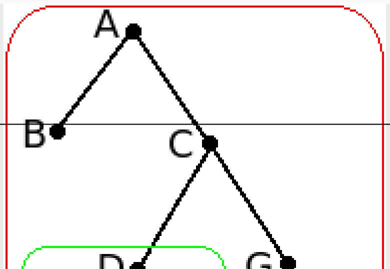
Additional options:

- db=dbName the rule will work only for the DB, the name of which matches the regular expression set by this parameter
- type=objectType the rule will work only for the objects with the specified type

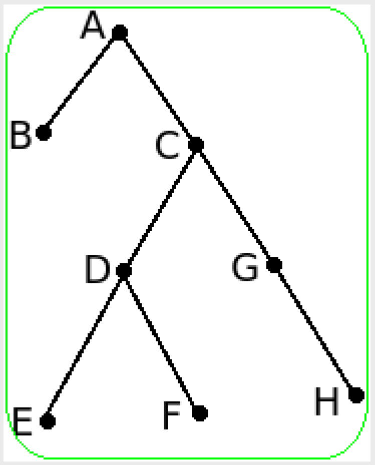
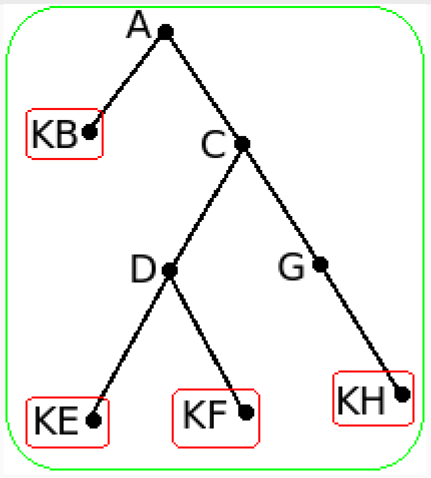
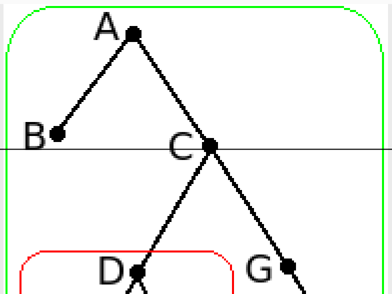
Note: To specify several types of one and the same object, you should separate them with commas.

Attention: Database name is case sensitive.

Examples for white list

Example and discription	Effective range
HIDE ALL [title] forbids displaying all the objects	
HIDE ALL [title] forbids displaying all the objects SHOW REGEX K [rule] allows to display objects that partially match the regular expression “K”	
	

Examples for black list

Example and discription	Effective range
SHOW ALL [title] allows displaying all the objects	
SHOW ALL [title] allows displaying all the objects HIDE REGEX K [rule] excludes objects which partially match the regular expression “K”	
	

Key words:

HIDE SHOW ALL REGEX CONTENT QUALIFIED NONE

These words cannot be identifiers, since they need to be enclosed in quotes to be used. Only the words which match the case perfectly, are reserved. For example - Content is an allowed identifier.

As an example of excluding the object - the name which matches perfectly (including the case) the key word SHOW:

```
SHOW ALL
HIDE NONE "SHOW"
```

20.2 Combined use of black and white lists

Black and white lits can be used together. In this case, their rules are combined into a common list. The rules which regulate displaying of one and the same object, combine into one common rule according to the following principles:

- if the rules are different in “range”, then the rule with a broader range (includes the object and its content) prevails
- if the “range” of the rules is the same, then the rule that hides the object prevails

Rule’s “range” is including or not including an object’s content into the effective range of the rule, that is, the condition of the CONTENT flag (for common list of the pgCodeKeeper graphic version, it’s the “Ignore contents” option, described in the Excluded objects section).

Example of combined use of black and white lists:

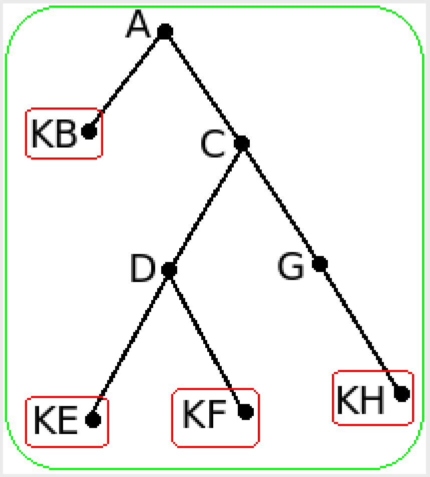
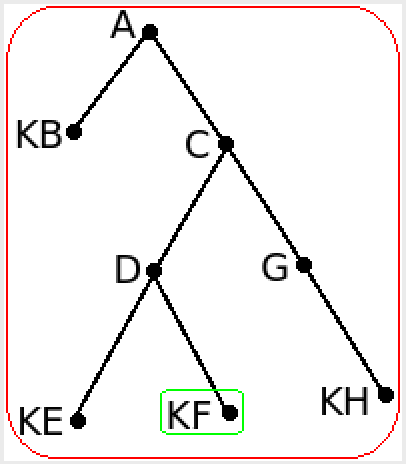
black list file

```
SHOW ALL
HIDE REGEX K
```

white list file

```
HIDE ALL
SHOW CONTENT KF
```

Effective range of the rule

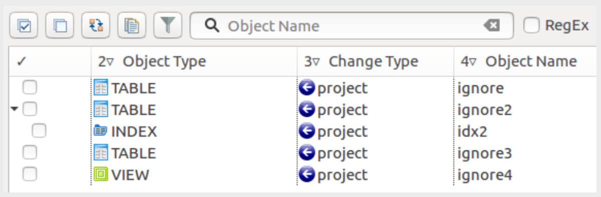
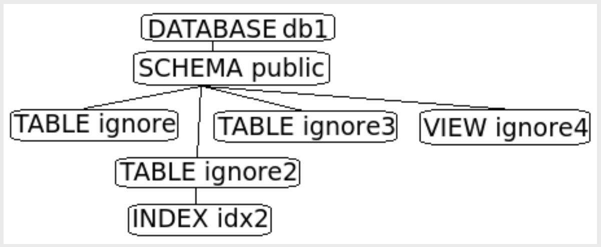
Black list	White List
	

As as result, the object with the name "KF" will be displayed, because the condition of the white list for current object surpasses the condition of the black list in "range".

Note: When working with the graphic version of pgCodeKeeper, adding the second list of exclusions is possible by using the general list of the excluded objects or by adding the external list through DB storage. CLI version pgCodeKeeper allows to add additional lists of exclusions with the help of the command: pgcodekeeper-cli --ignore-list <path> SOURCE DEST.

20.3 Examples of work with .pgcodekeeperignore

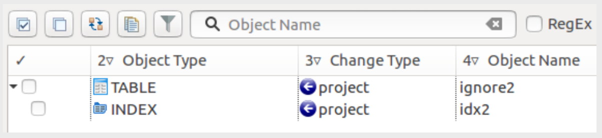
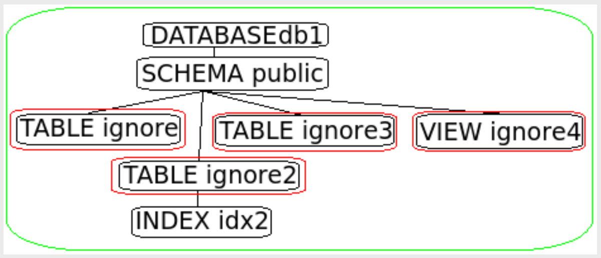
Suppose there is a view named ignore4 and a set of tables with the names: ignore, ignore2, ignore3. In its turn, ignore2 has some content.

Result	Schema
	

✓	2 Object Type	3 Change Type	4 Object Name
☐	TABLE	project	ignore
☐	TABLE	project	ignore2
☐	INDEX	project	idx2
☐	TABLE	project	ignore3
☐	VIEW	project	ignore4

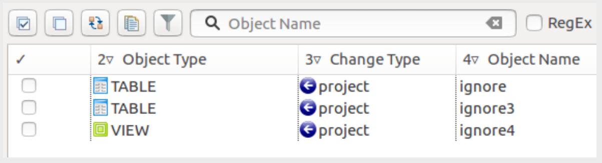
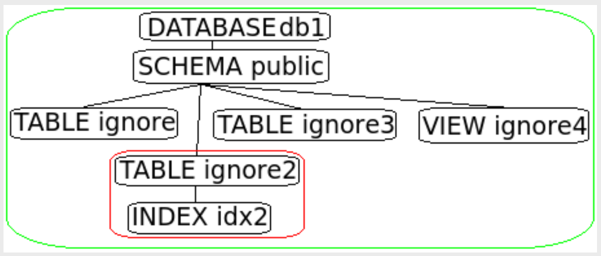
To exclude all the objects partially matching the regular expression "ignore", you should write the following rules in .pgcodekeeperignore:

```
SHOW ALL
HIDE REGEX ignore
```

Result	Schema
 Object Name · RegEx Object Type · Change Type · Object Name TABLE · project · ignore2 INDEX · project · idx2	 DATABASEdb1 SCHEMA public TABLE ignore TABLE ignore3 VIEW ignore4 TABLE ignore2 INDEX idx2

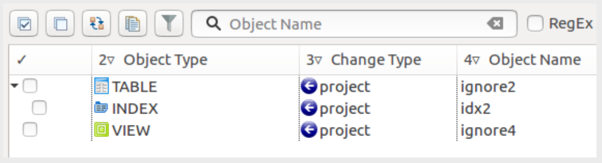
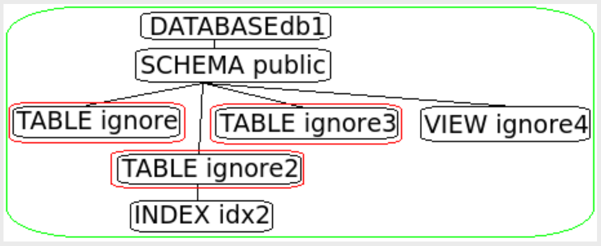
To exclude the object "ignore2" and its content, you should write the following rules in .pgcodekeeperignore:

```
SHOW ALL
HIDE CONTENT ignore2
```

Result	Schema
 Object Name · RegEx Object Type · Change Type · Object Name TABLE · project · ignore TABLE · project · ignore3 VIEW · project · ignore4	 DATABASEdb1 SCHEMA public TABLE ignore TABLE ignore3 VIEW ignore4 TABLE ignore2 INDEX idx2

To exclude all the objects with the "TABLE" type, partially matching the regular expression "ignore", you should write the following rules in .pgcodekeeperignore:

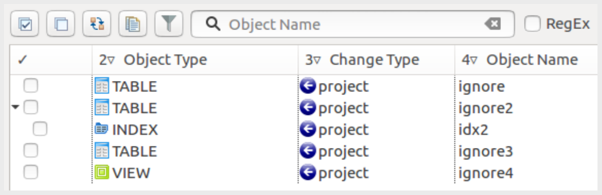
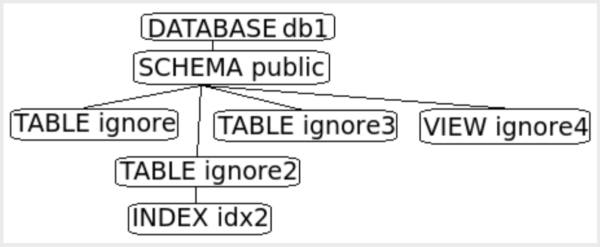
```
SHOW ALL
HIDE REGEX ignore type=TABLE
```

Result	Schema
 Object Name · RegEx Object Type · Change Type · Object Name TABLE · project · ignore2 INDEX · project · idx2 VIEW · project · ignore4	 DATABASEdb1 SCHEMA public TABLE ignore TABLE ignore3 VIEW ignore4 TABLE ignore2 INDEX idx2

To exclude all the objects partially matching the regular expression "ignore" with the "TABLE" type and content for the specified database, you should write the following rules in .pgcodekeeperignore:

```
SHOW ALL
HIDE CONTENT,REGEX ignore db=name_of_other_db type=TABLE
```

abovementioned rules use the name of another database, not the one which is in work in this example, so everything will remain unchanged

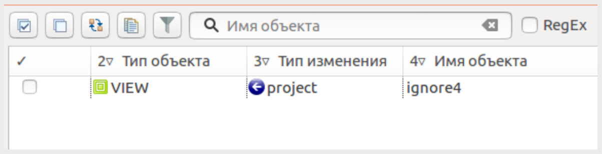
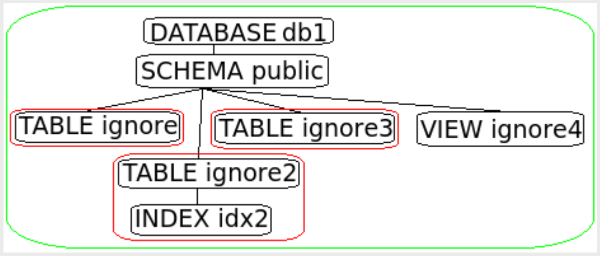
Result	Schema
	

Object Name RegEx

✓	2▿ Object Type	3▿ Change Type	4▿ Object Name
	TABLE	project	ignore
	TABLE	project	ignore2
	INDEX	project	idx2
	TABLE	project	ignore3
	VIEW	project	ignore4

but if you specify the name of the database in work, then all the objects will disappear from the displayed, except the one which doesn't match the type.

```
SHOW ALL
HIDE CONTENT,REGEX ignore db=db1 type=TABLE
```

Result	Schema
	

Имя объекта RegEx

✓	2▿ Тип объекта	3▿ Тип изменения	4▿ Имя объекта
	VIEW	project	ignore4

To allow displaying the object "ignore2" using black and white lists at the same time, you need to write the following rules:

file .pgcodekeeperignore - black list

```
SHOW ALL
HIDE REGEX ignore
```

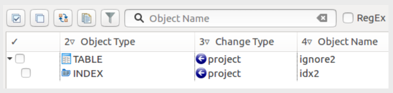
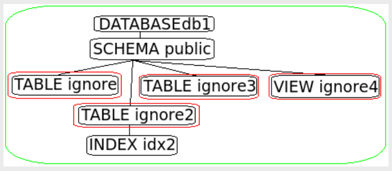
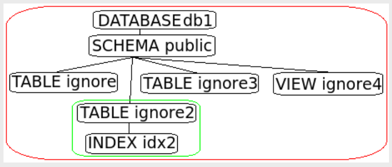
additional file of the list of excluded objects - white list

```
HIDE ALL
SHOW CONTENT ignore2
```

Note: How to add an additional list is discribed in the Combined use of black and white lists section.

The aim of the black list is to exclude all the objects partially matching the regular expression "ignore".

The aim of the white list is to remove "ignore2" from the excluded objects.

Result	Black list	White List
 Object Name RegEx ✓ | 2 Object Type | 3 Change Type | 4 Object Name TABLE | project | ignore2 INDEX | project | idx2	 DATABASEdb1 SCHEMA public TABLE ignore TABLE ignore3 VIEW ignore4 TABLE ignore2 INDEX idx2	 DATABASEdb1 SCHEMA public TABLE ignore TABLE ignore3 VIEW ignore4 TABLE ignore2 INDEX idx2

As a result, only "ignore2" will remain in the compared objects, because thanks to the "CONTENT" flag for the "ignore2" the white list rule surpasses the black list rule "HIDE REGEX ignore" in "range".

Note: Interaction between the rules of different lists which regulate the display of one and the same object is described in the Combined use of black and white lists section.

Глава

TWENTYONE

IGNORING SCHEMAS WHEN DOWNLOADING

pgCodeKeeper has a feature of a selective downloading of objects from the database. To download the content of specific database schemas, you should select them on the list of ignored schemas.

List of ignored schemas is a list of schemas and their content that shouldn't or should be downloaded from the DB.

There are two types of lists of ignored schemas:

black list
: allows displaying of all the objects except the ones from the specified schemas and the schemas themselves in the list rules;

white list
: as opposed to the black list, it forbids displaying of all the objects except the ones from the specified schemas and the schemas themselves in the list rules.

In each project, it is possible to create an ignored objects list, applying only to current project. To do this, create the .pgcodekeeperignoreschema file in the project root and fill it with the rules according to the list syntax.

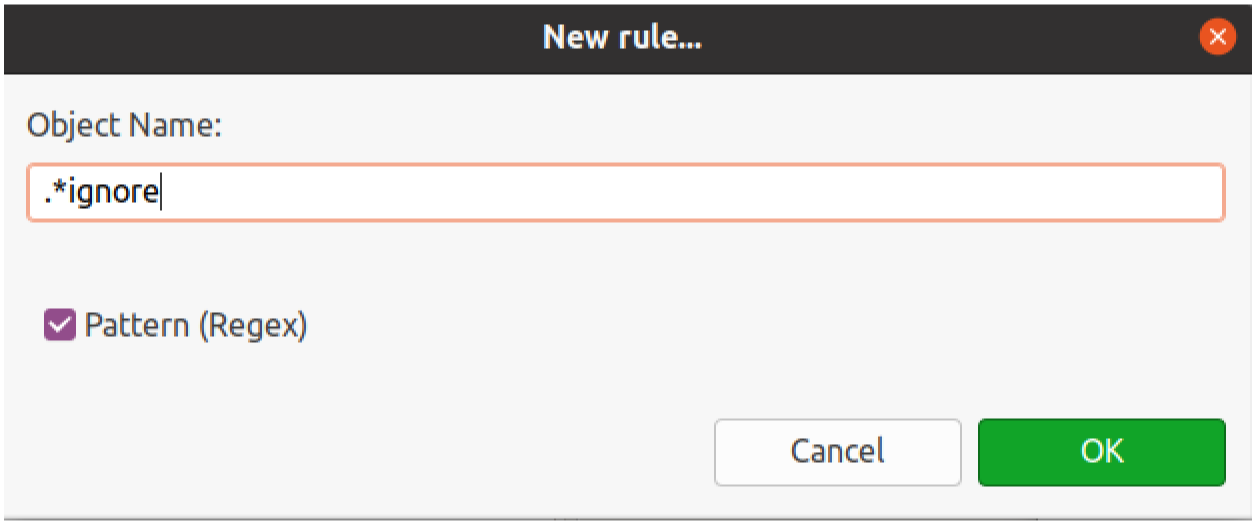
> Attention: The .pgcodekeeperignoreschema file format is case-sensitive. You should use UTF-8.

For the .pgcodekeeperignoreschema file, the same rules are applied, as for Ignore List (see the List syntax section).

We can apply the list of ignored schemas in DB storage. CLI version pgCodeKeeper with the help of the following command: pgcodekeeper-cli --ignore-schema <path> SOURCE DEST.

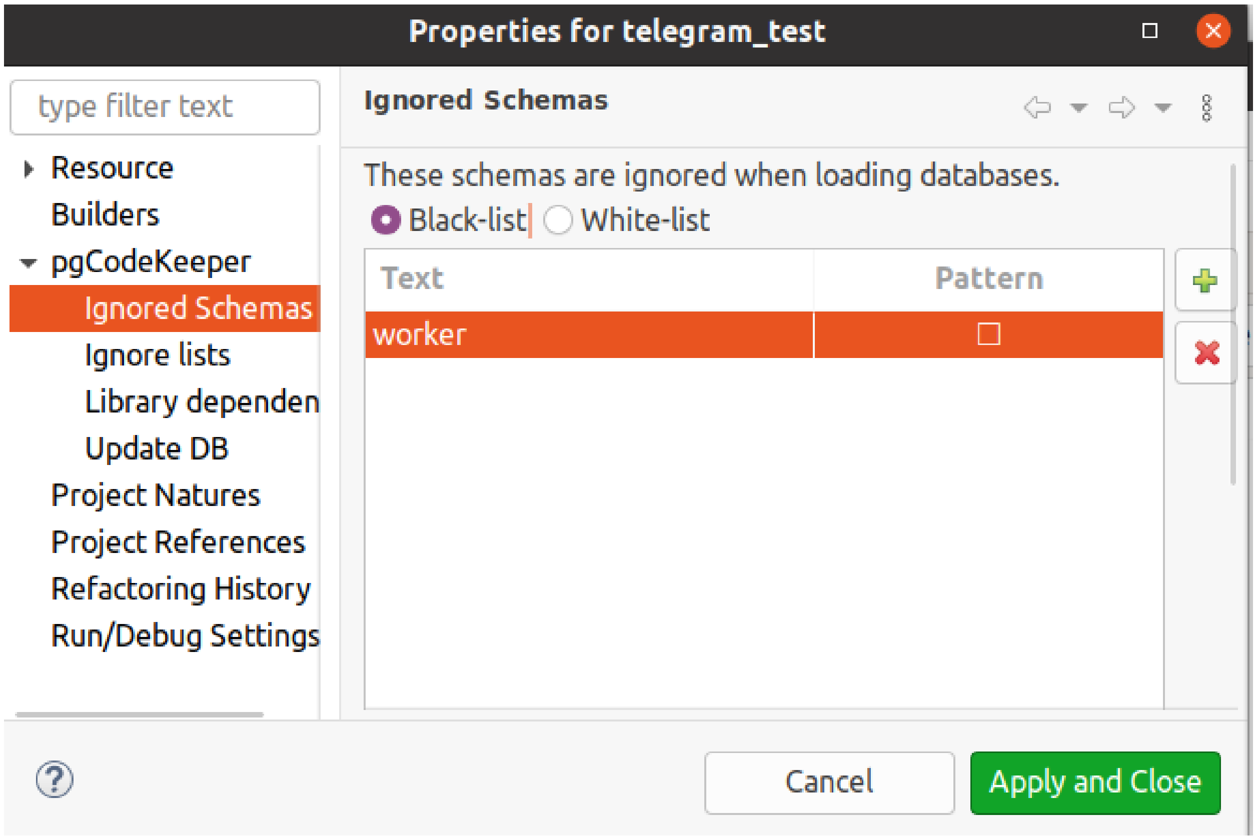
21.1 Object selection

If you tick Pattern(REGEX) in the new rule creation window, objects will be searched by a regular expression specified in the object name field.



21.2 Example of using the ignoring of schemas in downloading

For instance, we need to download all the database objects, except for the objects from the worker schema. Add worker to the list of the ignored schemas, tick Black-list on the interface of the ignored schemas:



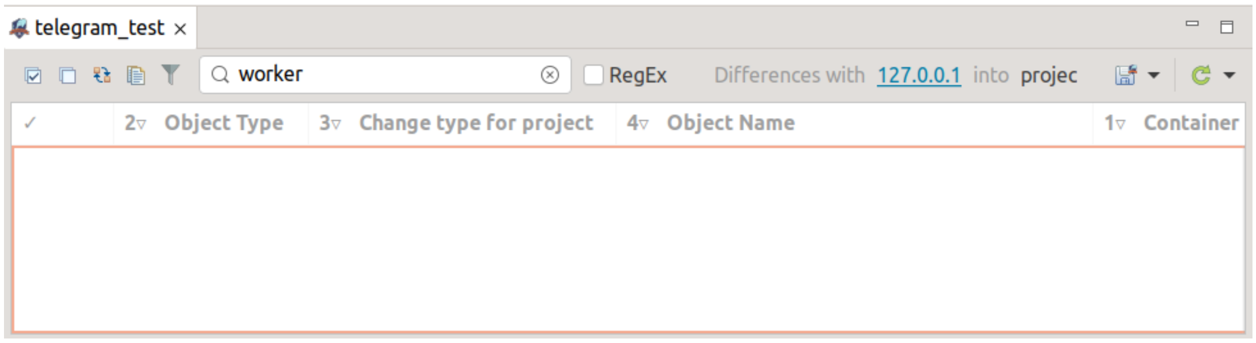
These schemas are ignored when loading databases.

[✓ Black-list] White-list

Text	Pattern
worker	☐

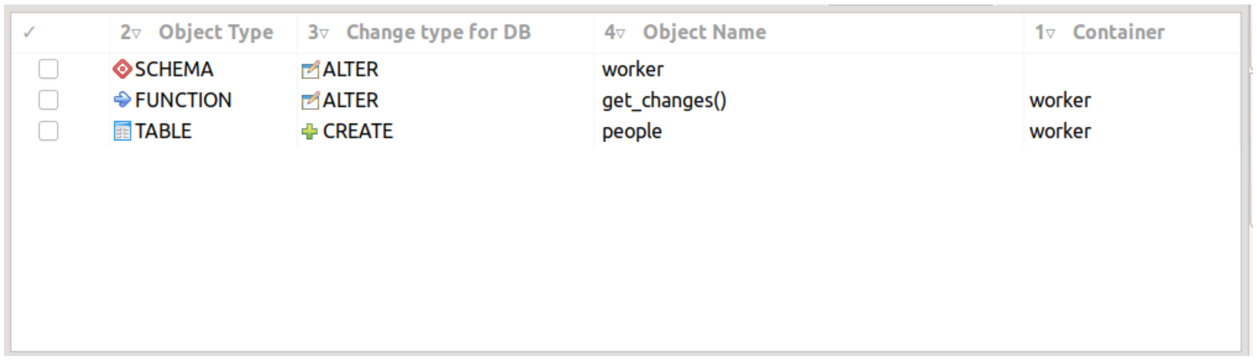
Click Apply and Close.

In the project editor, click Changes . There are no objects from the worker schema on the differences panel!



✓	2▿ Object Type	3▿ Change type for project	4▿ Object Name	1▿ Container

If you need to download the objects from the worker schema only, tick White-list in the ignored schemas interface. As a result, only the objects from the worker schema will be displayed.



✓	2▿ Object Type	3▿ Change type for DB	4▿ Object Name	1▿ Container
☐	SCHEMA	ALTER	worker	
☐	FUNCTION	ALTER	get_changes()	worker
☐	TABLE	CREATE	people	worker

Глава

TWENTYTWO

CLI VERSION

As an alternative for Eclipse IDE plug-in, it is possible to compare database schemas by using command line. To do this, you should download pgCodeKeeper-cli application, which works in stand-alone mode.

Console build of the program can be used for:

- creating a pgCodeKeeper "project" from file structure dump (without Eclipse project)
- creating migration scripts using pgCodeKeeper dumps and file structures
- creating migration scripts using database connections
- viewing the dependences of objects

22.1 Getting Started

First off, install Java SDK for your platform.

The latest version of cli build can be downloaded here. In the unpacked archive use files for parameter passing: pgcodekeeper-cli.sh for Linux systems and pgcodekeeper-cli.bat for Windows systems.

You can view the set of parameters available for work with the program by executing the command:

```
./pgcodekeeper-cli.sh --help
```

22.2 Examples

Writing migration script of the differences from the project of the test1 directory into the testdb1 database on the localhost:5432 server into the diff.sql file:

```
./pgcodekeeper-cli.sh -o /home/codekeeper/projects/diff.sql 'jdbc:postgresql://localhost:5432/testdb1?
↪user=user&password=password' /home/codekeeper/projects/test1/
```

Entering migration script from the backup file in the project of the test1 directory into the console:

```
./pgcodekeeper-cli.sh /home/codekeeper/backup.sql /home/codekeeper/projects/test1/
```

Creation of the project in the test1 directory basing on the testdb1 database on the localhost:5432 server:

```
./pgcodekeeper-cli.sh --parse -o /home/codekeeper/projects/test1/ 'jdbc:postgresql://localhost:5432/
↪testdb1?user=user&password=password'
```

Displaying the dependences tree of the testdb1 database on the localhost:5432 server:

```
./pgcodekeeper-cli.sh --graph 'jdbc:postgresql://localhost:5432/testdb1?user=user&password=password'
```

Displaying the tree of objects on which t1 and t2 objects depend in the testdb1 database on the localhost:5432 server:

```
./pgcodekeeper-cli.sh --graph --graph-reverse --graph-name t1 --graph-name t2 'jdbc:postgresql://
↪localhost:5432/testdb1?user=user&password=password'
```

All parameters after -vmargs will be transferred to VM.

Usage with restrictions of the memory used:

```
./pgcodekeeper-cli.sh 1.sql 2.sql -vmargs -Xms256m -Xmx2g
```

The VM parameter ru.taximaxim.codekeeper.parser.poolsize allows specifying the number of parser threads:

```
./pgcodekeeper-cli.sh 1.sql 2.sql -vmargs -Dru.taximaxim.codekeeper.parser.poolsize=5
```

Глава

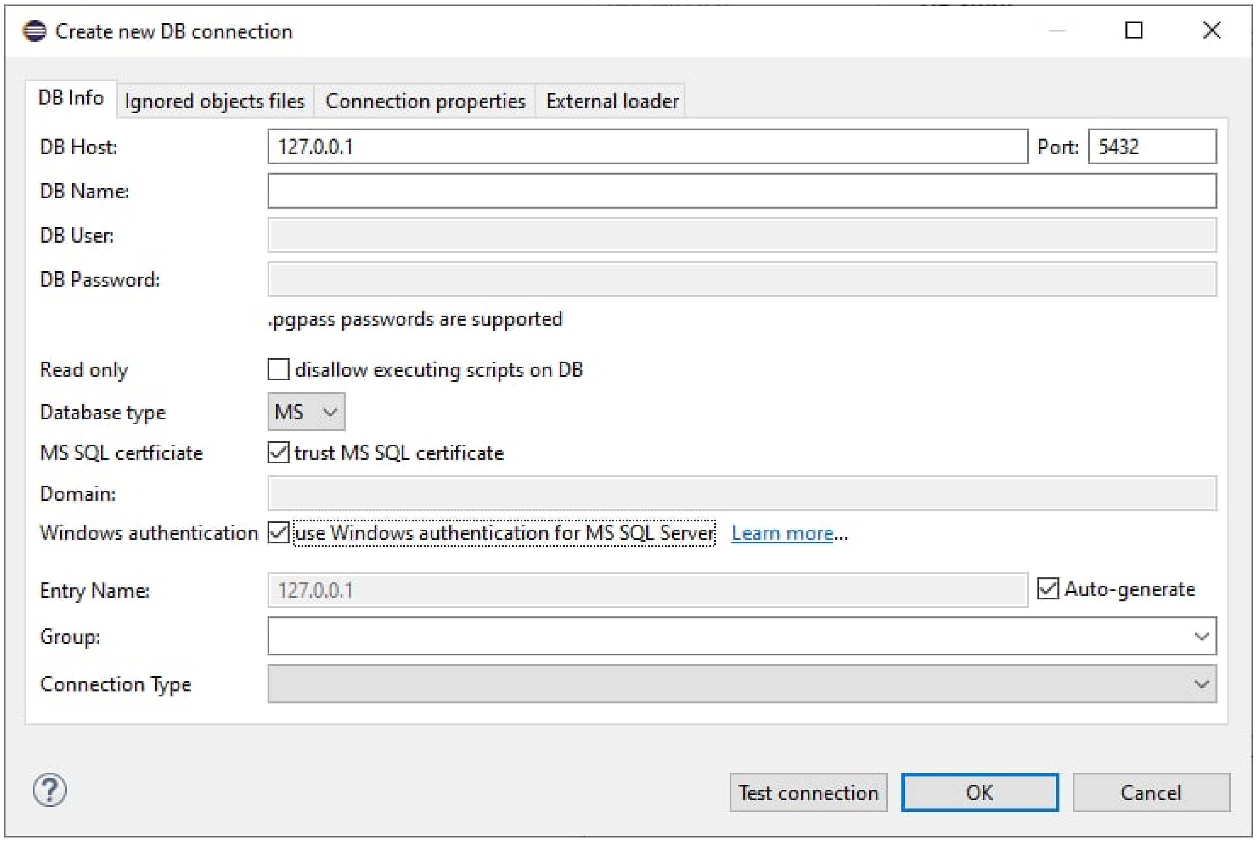
TWENTYTHREE

WINDOWS AUTHENTICATION

When working with Windows, you can use the Windows authentication mode to connect to the Microsoft SQL DB servers. This mode allows you to use the data of the current Windows user or the user you run a program as, rather than explicitly indicating the username and password.

To enable the Windows authentication mode, you just need to enable the corresponding option in DB connection settings. If you use a CLI version, you'll have to indicate the integratedSecurity parameter in connection string.

Examples of using Windows authentication in GUI and CLI:



Create new DB connection

DB Info | Ignored objects files | Connection properties | External loader

DB Host: 127.0.0.1 Port: 5432

DB Name:

DB User:

DB Password:

.pgpass passwords are supported

Read only: [ ] disallow executing scripts on DB

Database type: MS

MS SQL certficiate: [✓] trust MS SQL certificate

Domain:

Windows authentication: [✓] use Windows authentication for MS SQL Server Learn more...

Entry Name: 127.0.0.1 [✓] Auto-generate

Group:

Connection Type

```
pgcodekeeper-cli.bat --ms-sql jdbc:sqlserver://127.0.0.1;databaseName={master};integratedSecurity=true␣
↪NUL
```

23.1 System configuration

To use Windows authentication, the app must have access to the mssql-jdbc_auth Microsoft JDBC driver DLL of the appropriate version. Download the DLL file from the release pack and put it to the directory present in the PATH environment variable, or in the java.library.path JVM property. Library bitness should match the bitness of the Eclipse version you use.

The easiest and most universal way to achieve this is to save the library in the C:\Windows\System32\ directory. In 64-bit systems, this directory contains 64-bit libraries. If you use a 32-bit Eclipse version for the 64-bit system, save the 32-bit library in the C:\Windows\SysWOW64\ directory.

Глава

TWENTYFOUR

CONTRIBUTION OF THE PARTICIPANTS

This product is created with participation of many people. Here you'll see the people who contributed to the product development.

24.1 Concept

- Sergei Mokeev

Dmitry Novgorodov's and Andrei Volodin's contribution to the shaping of concepts and ideas deserves an honorable mention as well.

24.2 Management

- Andrew Saushkin - product and sometimes project management

24.3 Development

- Alexander Levsha
- Mansur Galiev
- Lenar Shamsutdinov
- Anatoly Botov
- Anton Ryabinin
- Anton Akifiev
- Gulnaz Khazieva
- Mikhail Tavturin
- Ildar Shaydullin
- Danil Sikorsky
- Elmir Shamsutdinov

24.4 Website design and layout

- Gulnaz Khazieva
- Vasily Demakov

24.5 Documentation Translation

- Kirill Pigin
- We are in search of contributors who would translate the documentation.

Глава

TWENTYFIVE

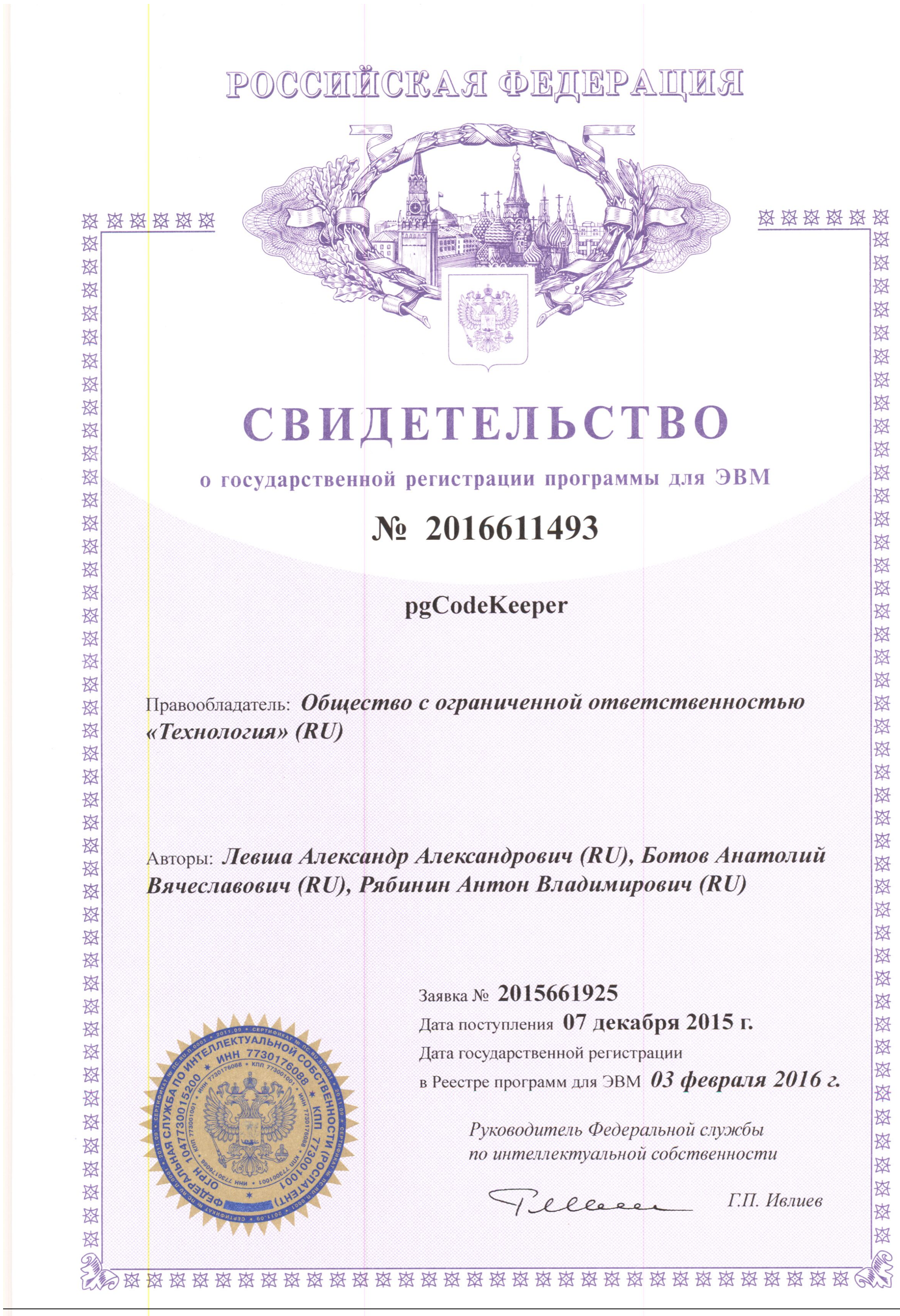
ABOUT PGCODEKEEPER

The pgCodeKeeper program is developed and maintained by Taxtelecom LLC (formerly Technology LLC).

25.1 Organization card

Full name	Limited liability company Taxtelecom
Abbreviated name	Taxtelecom LLC
Primary State Registration Number	1114501006652
TIN/RRC	4501170000/720301001
Legal address	ul. Proletarskaya, d.39, kv.43, g.o. Kurgan city, Kurgan, Kurgan oblast', Russia, 640018
Mailing Address	ul. Radionova, d. 17, city of Kurgan, Kurganskaya oblast, 640003
Phone	+7 (3522) 63-03-40
Fax	+7 (3522) 60-14-09
OKPO	87299772
Director	Tatarintsev Igor Vitalyevich

25.2 Registration certificate



РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016611493

pgCodeKeeper

Правообладатель: ***Общество с ограниченной ответственностью «Технология» (RU)***

Авторы: ***Левша Александр Александрович (RU), Ботов Анатолий Вячеславович (RU), Рябинин Антон Владимирович (RU)***

Заявка № **2015661925**

Дата поступления **07 декабря 2015 г.**

Дата государственной регистрации

в Реестре программ для ЭВМ ***03 февраля 2016 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

[illegible] *Г.П. Ивлиев*

ФЕДЕРАЛЬНАЯ СЛУЖБА ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ (РОСПАТЕНТ) ОГРН 1047730015200 ★ ИНН 7730176088 ★ КПП 773001001
СЕРТИФИКАТ № ПС.RU.П.0001 • 2011.09
ИНН 7730176088 • КПП 773001001