Article
Yuri Marx · Jan 25, 2022 4m read

Data migration tool - Part I: from Postgres to IRIS

Sometimes it is necessary to transfer or migrate data and data schema from Postgres to IRIS. There are currently a few options for doing this, but the two most popular options are using DBeaver (https://openexchange.intersystems.com/package/DBeaver) or SQLGateway. The first will be demonstrated in this article and the second is presented in an excellent article by Robert Cemper, DB Migration using SQLgateway (https://community.intersystems.com/post/db-migration-using-sqlgateway), see in this article how to perform this migration using DBeaver:

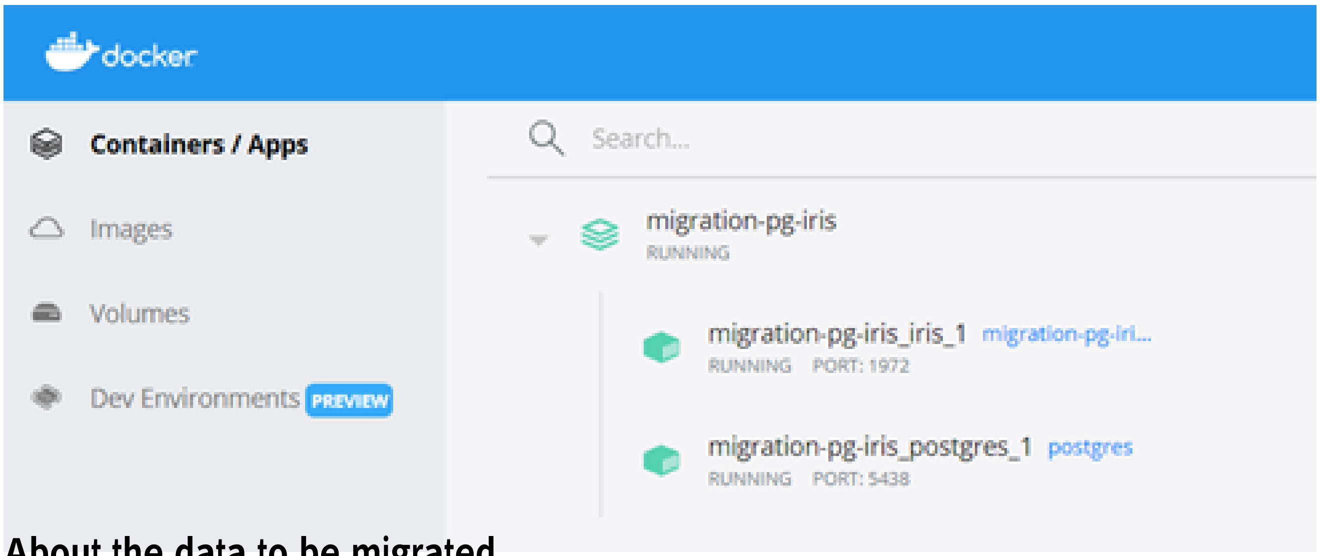
Get the sample data to the migration process

In the Github is possible download a docker compose project to build and run 2 databases:

- **Source Database**: PostgreSQL database Docker instance with a sample database.
- **Target Database**: InterSystems IRIS data platform Docker instance with a ready schema to receive the source database.

To get the sample and run it, follow these steps:

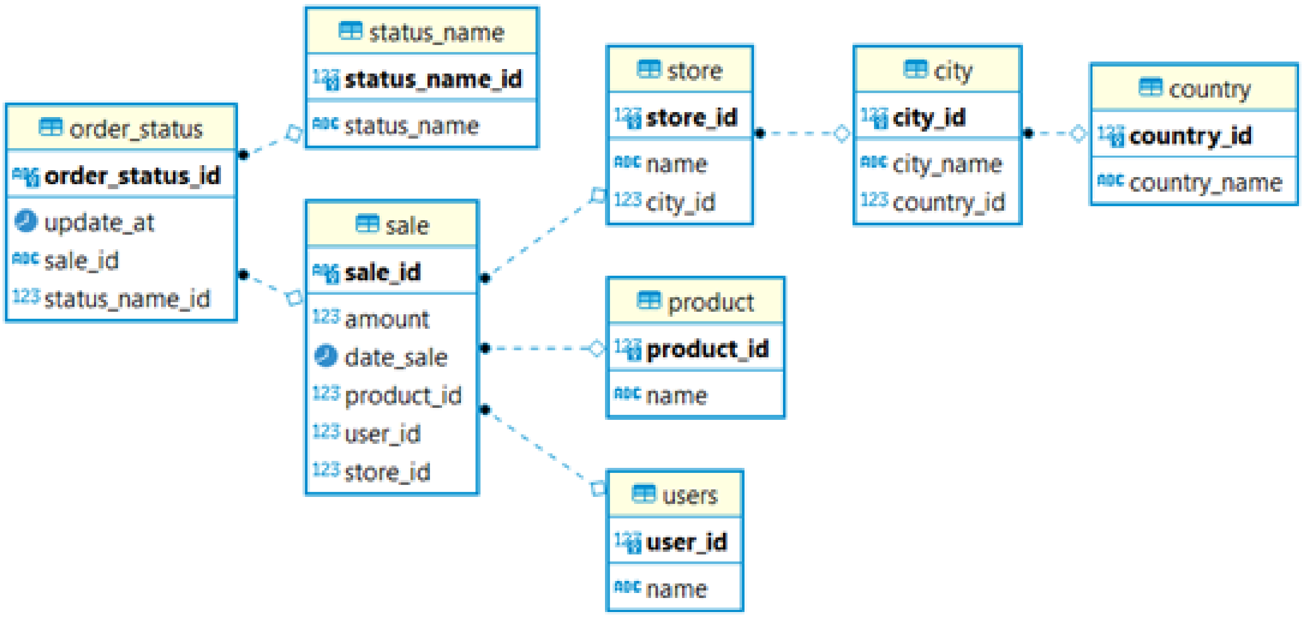
1. Go to the https://github.com/yurimarx/migration-pg-iris and click Download to go to the git repository.
2. Clone the project: git clone https://github.com/yurimarx/migration-pg-iris.git.
3. Go to the project folder migration-pg-iris.
4. Do the build: docker-compose build.
5. Execute the containers: docker-compose up -d.
6. See in your docker desktop with the instances are ok:



migration-pg-iris
RUNNING
migration-pg-iris_iris_1 migration-pg-iri...
RUNNING PORT: 1972
migration-pg-iris_postgres_1 postgres
RUNNING PORT: 5438

About the data to be migrated

The data to be migrated is represented here:



So, the migration process from PostgreSQL to IRIS will migrate:

- 08 tables.
- 1000000 rows of sale.
- 250000 rows of users.
- 300 rows of product.
- 500 rows of store.
- 100 rows of country.
- 30 rows of city.
- 5 rows of statusname.

The migration destination will be dctest schema inside USER namespace in the InterSystems IRIS database.

Open-source tool to migrate from PostgreSQL to IRIS: DBeaver

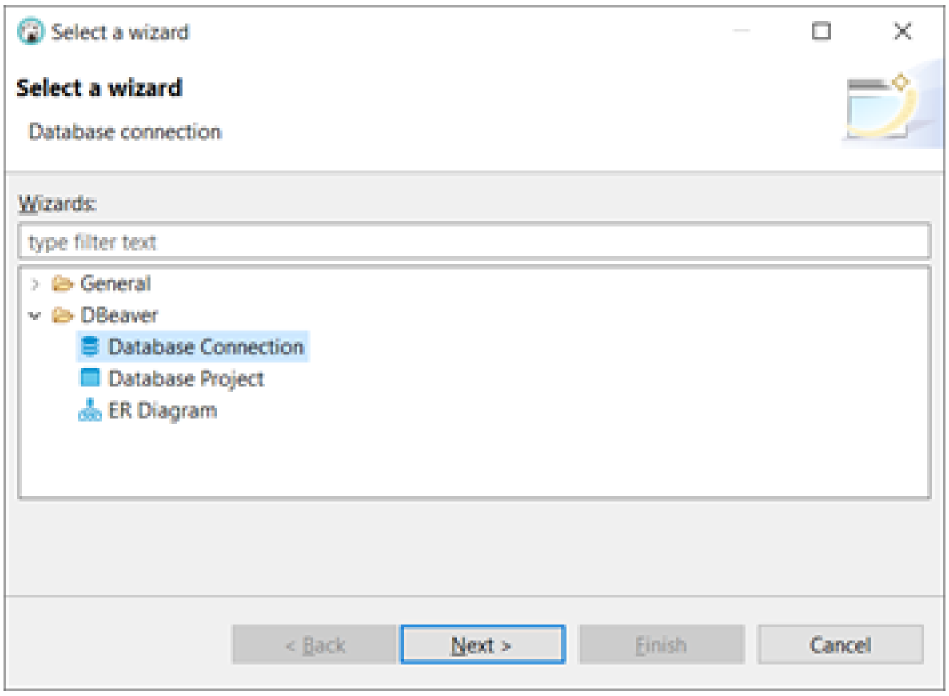
DBeaver is a database tool to connect, create, drop, select, update and delete data objects to the main database products in the market. Download it from: https://openexchange.intersystems.com/package/DBeaver. Now follow the installation instructions to get this fantastic product into your laptop or desktop.
DBeaver can be used to migrate data between database connections, even if they are from different manufacturers and versions.

Connecting the Source and Target Databases using the DBeaver

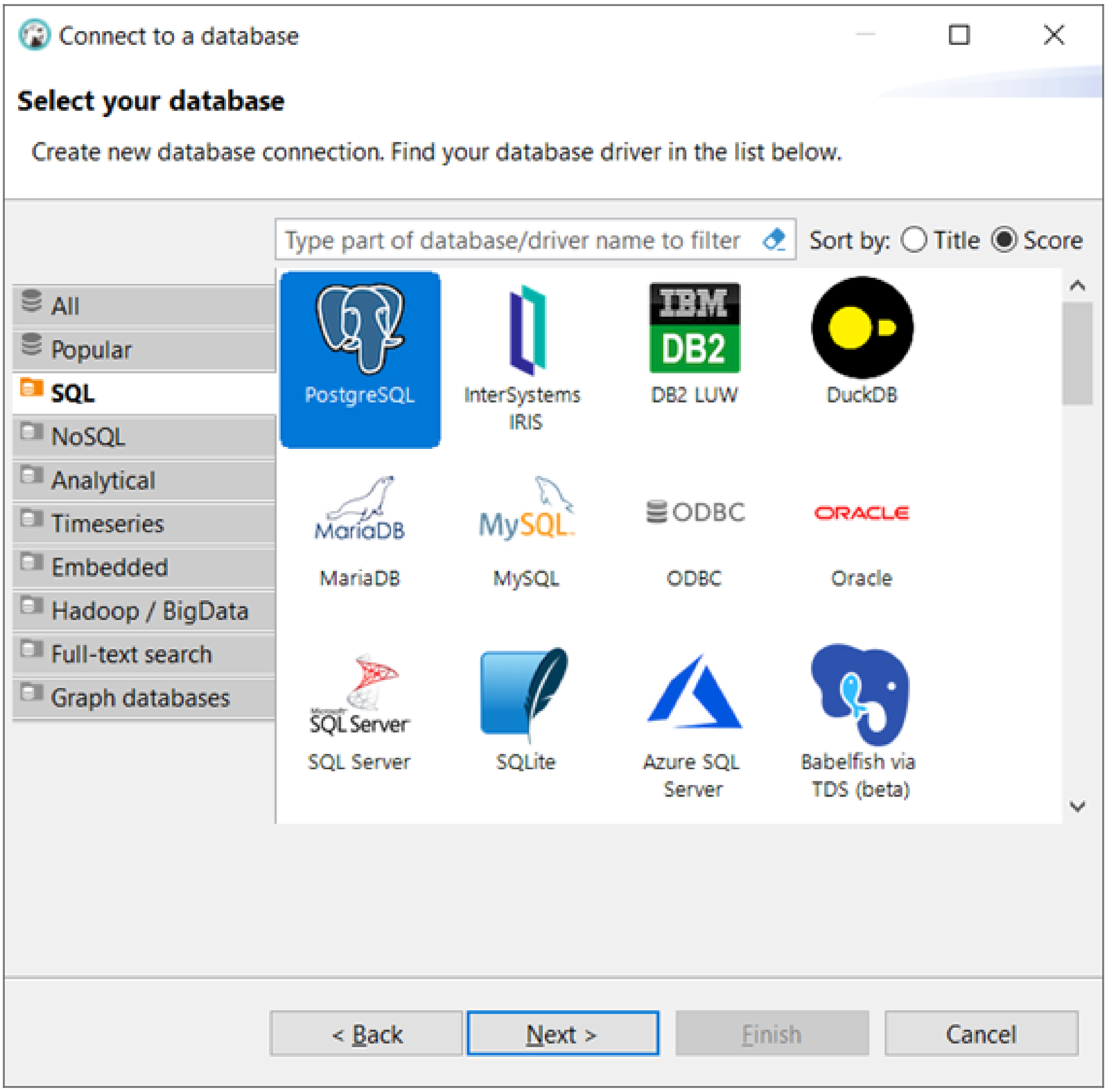
Now we will set the database connections to be migrated.
To set PostgreSQL connection to the DBeaver:

1. In the DBeaver Go to File > New.

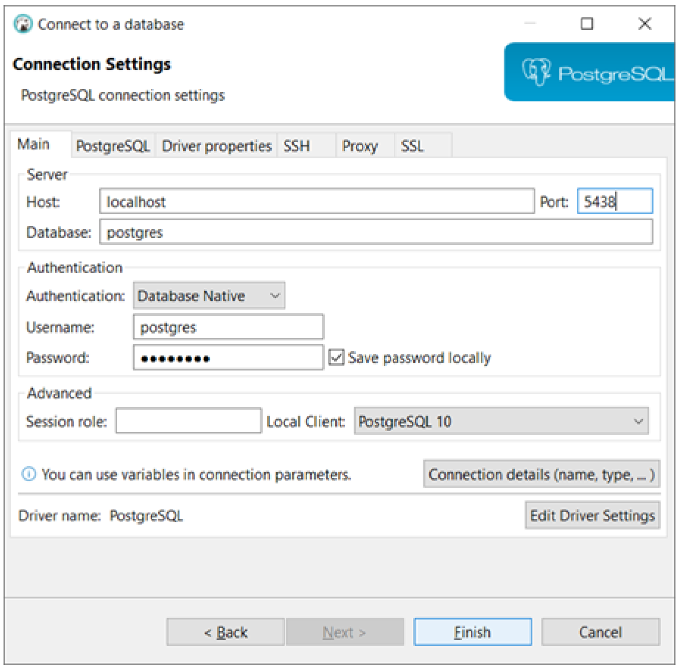
2. Select Database Connection and click Next:



3. Choose SQL tab > PostgreSQL and click next:



4. Fill the PostgreSQL connection fields like this picture:
    - Host: localhost
    - Port: 5438
    - Database: postgres
    - Username: postgres
    - Password: postgres
    - Click Finish.



Connection Settings

PostgreSQL connection settings

Server

Host: localhost Port: 5438

Database: postgres

Authentication

Authentication: Database Native

Username: postgres

Password: •••••••• ☑ Save password locally

Advanced

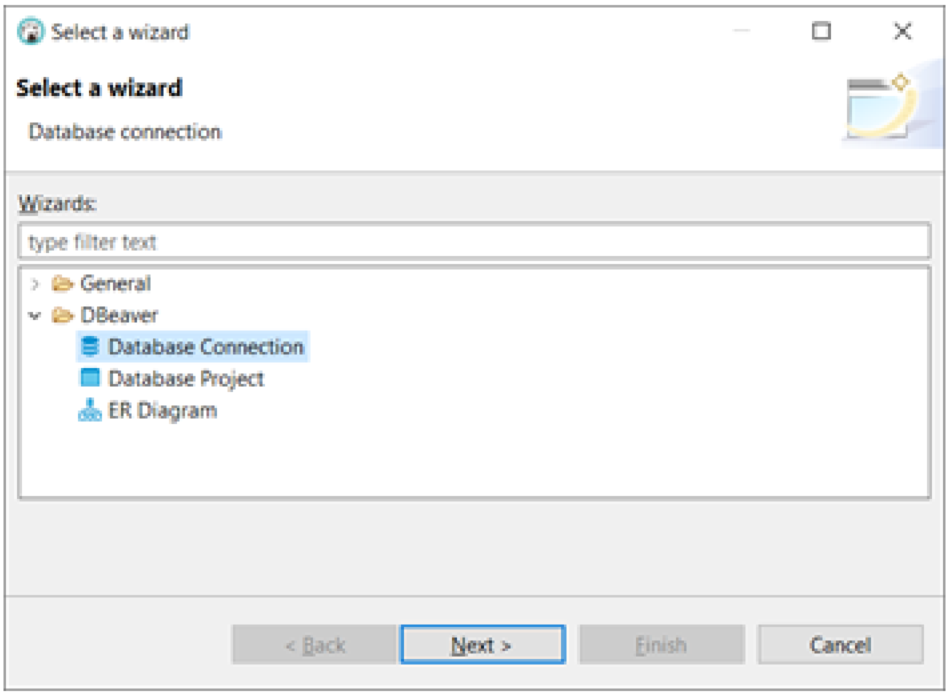
Session role: Local Client: PostgreSQL 10

You can use variables in connection parameters.

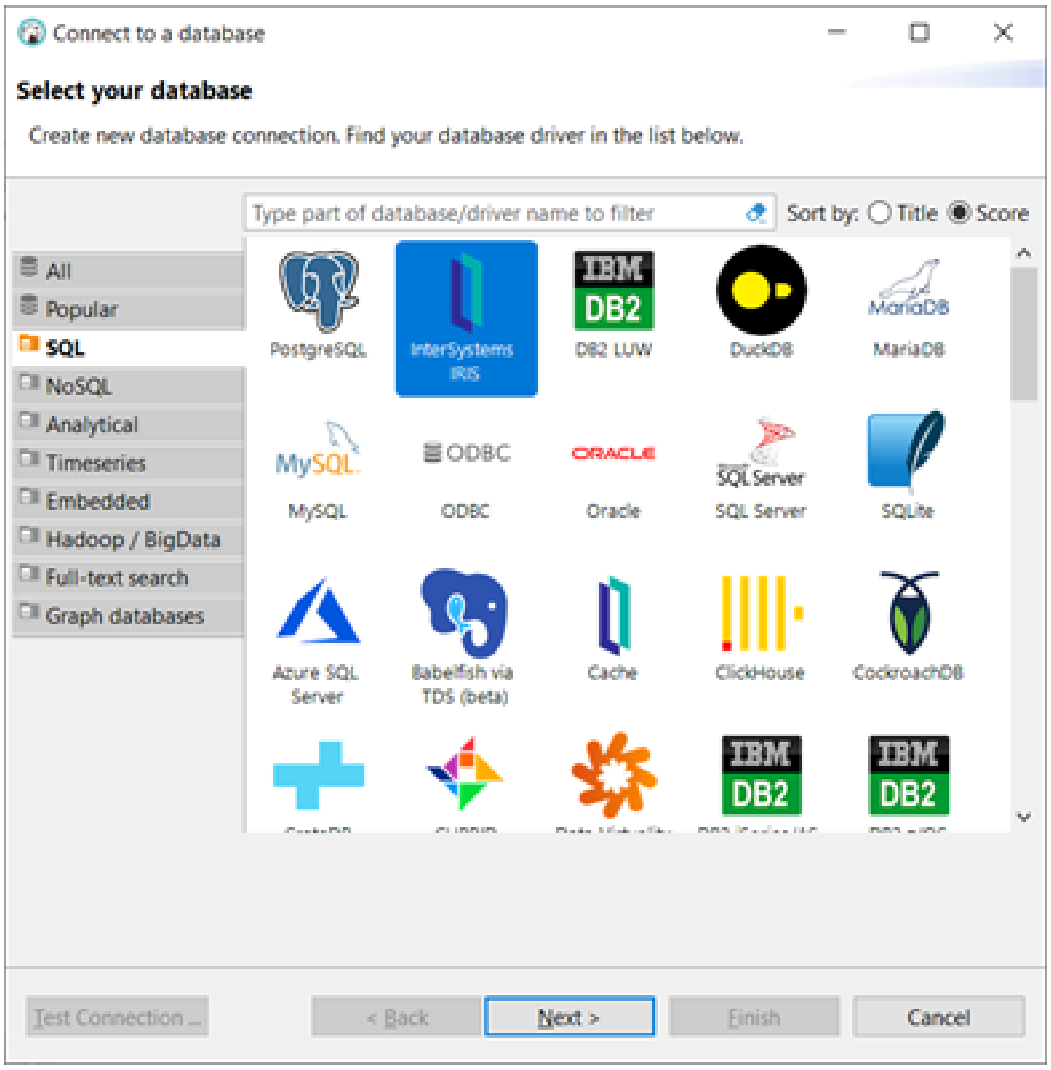
Driver name: PostgreSQL

To set InterSystems IRIS connection to the DBeaver:

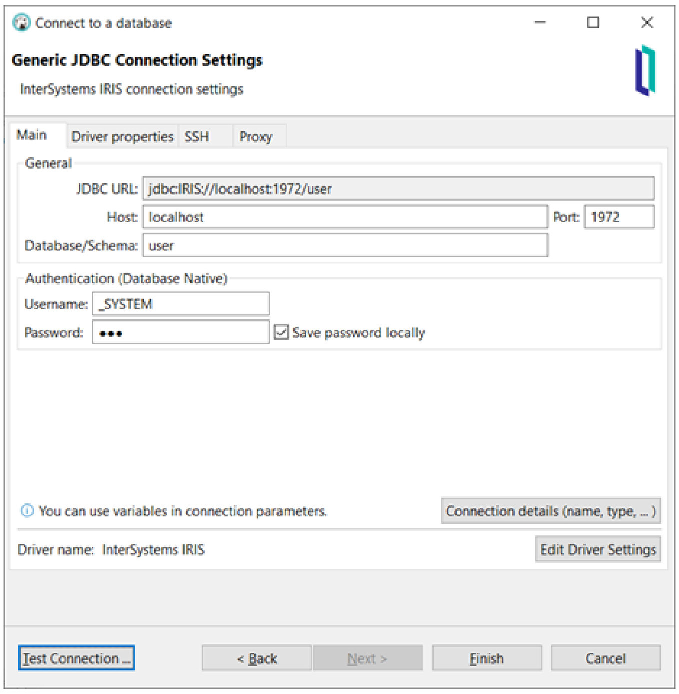
1. In the DBeaver Go to File > New.
2. Select Database Connection and click Next:



3. Choose SQL tab > InterSystems IRIS and click next:



4. If DBeaver requests to download the InterSystems IRIS driver, press Yes or Ok.
5. Set the InterSystems IRIS connection fields like this picture:
   - Host: localhost
   - Database/Schema: user
   - Username: _SYSTEM
   - Password: SYS
   - Click Text Connection and Finish.


Connect to a database

Generic JDBC Connection Settings

InterSystems IRIS connection settings

Main | Driver properties | SSH | Proxy

General

JDBC URL: jdbc:IRIS://localhost:1972/user

Host: localhost Port: 1972

Database/Schema: user

Authentication (Database Native)

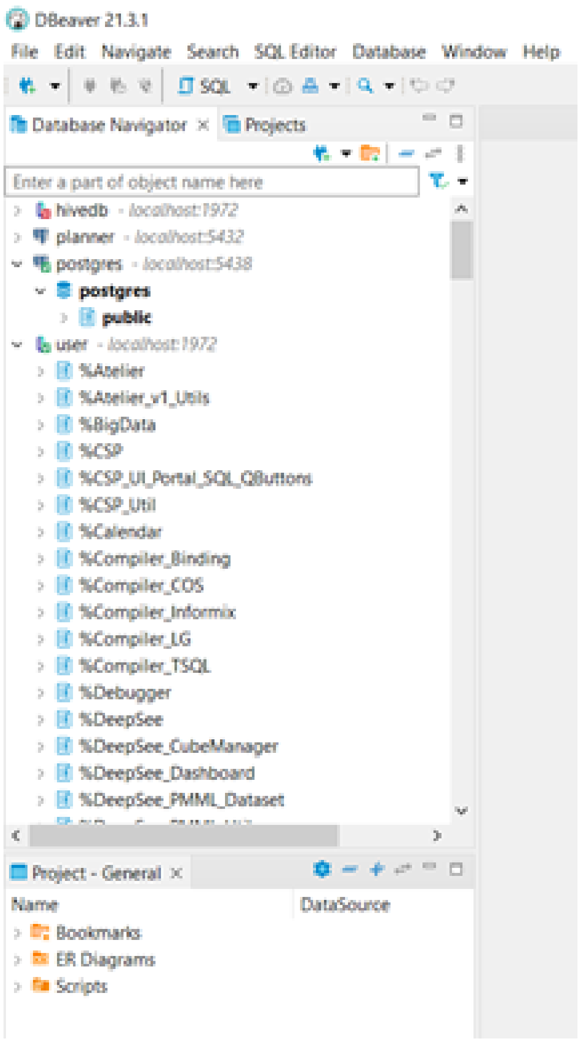
Username: _SYSTEM

Password: ••• ☑ Save password locally

You can use variables in connection parameters.

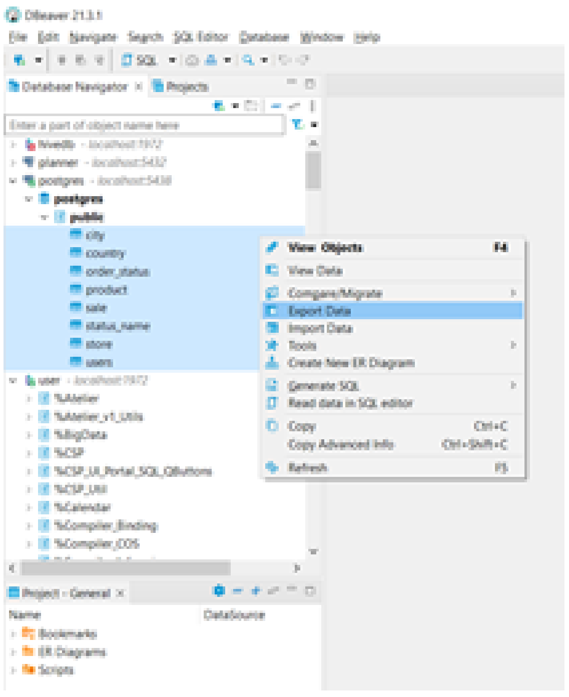
Driver name: InterSystems IRIS

6. The connections (postgres and user) are available in the Database Navigator:

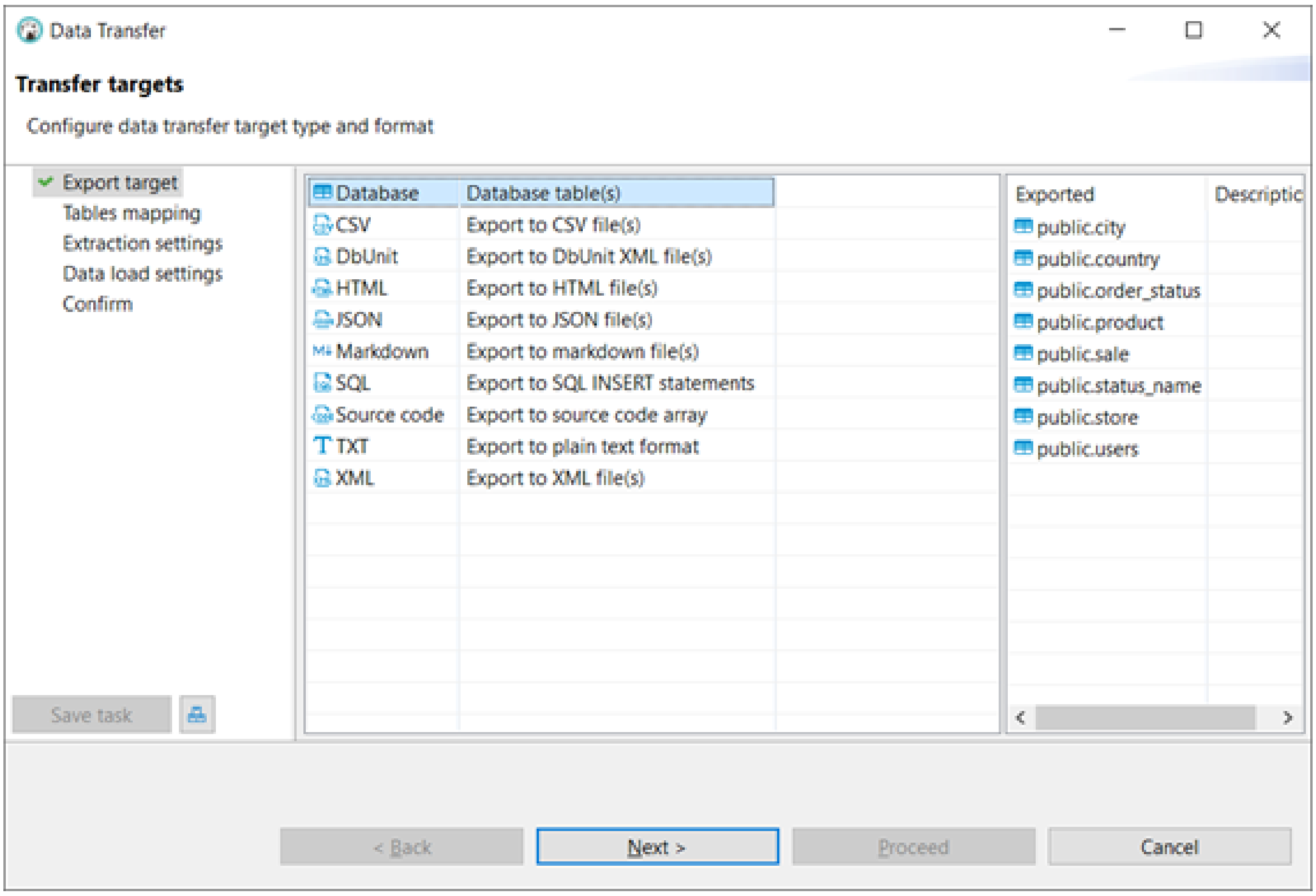


Do the migration

To do the migration, follow these steps:
1. Expand the postgres connection > public and select all tables. Click with the mouse right button with the tables selected and choose Export Data, like this picture:



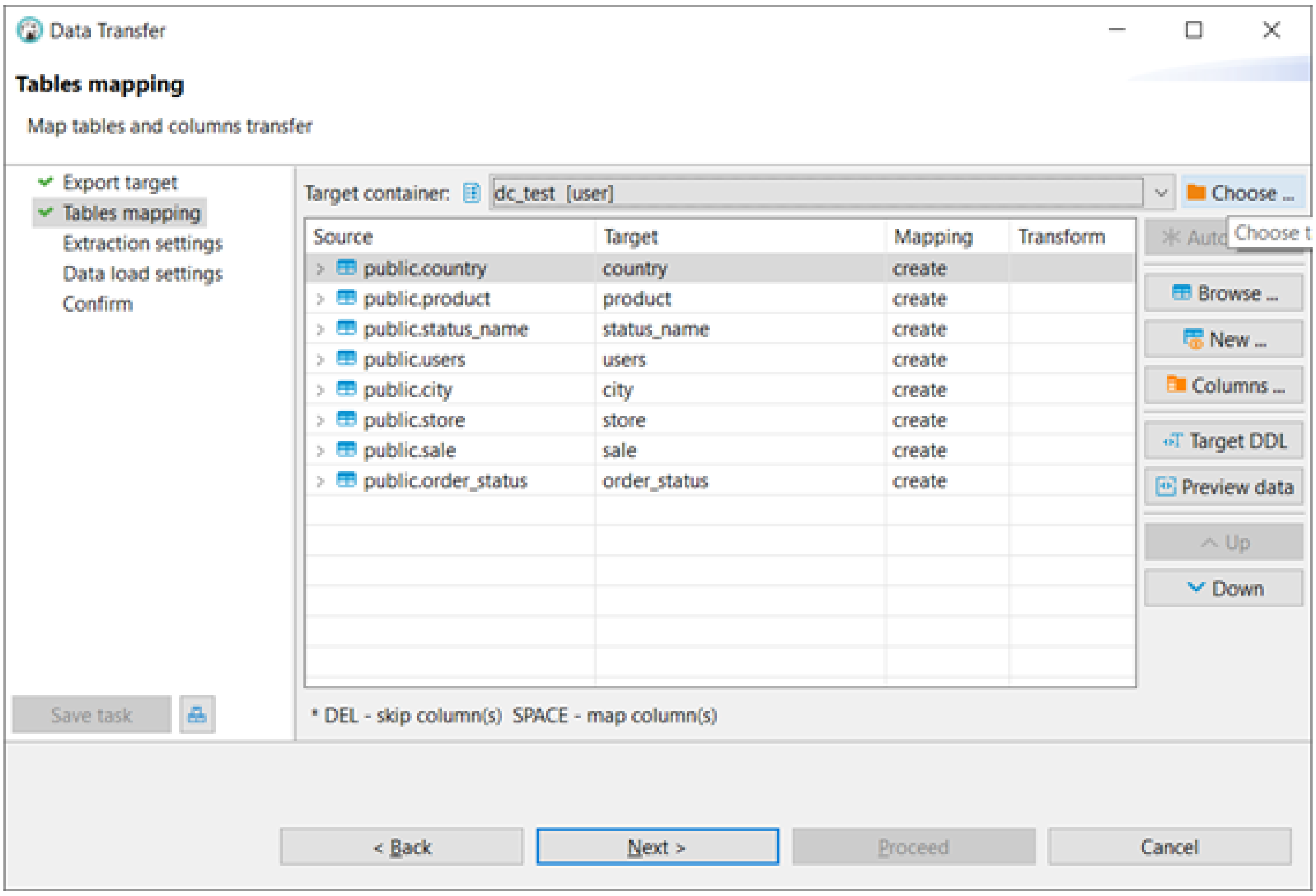
2. Select Database, like in this picture and click Next



Database	Database table(s)
CSV	Export to CSV file(s)
DbUnit	Export to DbUnit XML file(s)
HTML	Export to HTML file(s)
JSON	Export to JSON file(s)
Markdown	Export to markdown file(s)
SQL	Export to SQL INSERT statements
Source code	Export to source code array
TXT	Export to plain text format
XML	Export to XML file(s)

Exported	Descriptic
public.city	
public.country	
public.order_status	
public.product	
public.sale	
public.status_name	
public.store	
public.users	

3. Click the Choose button:

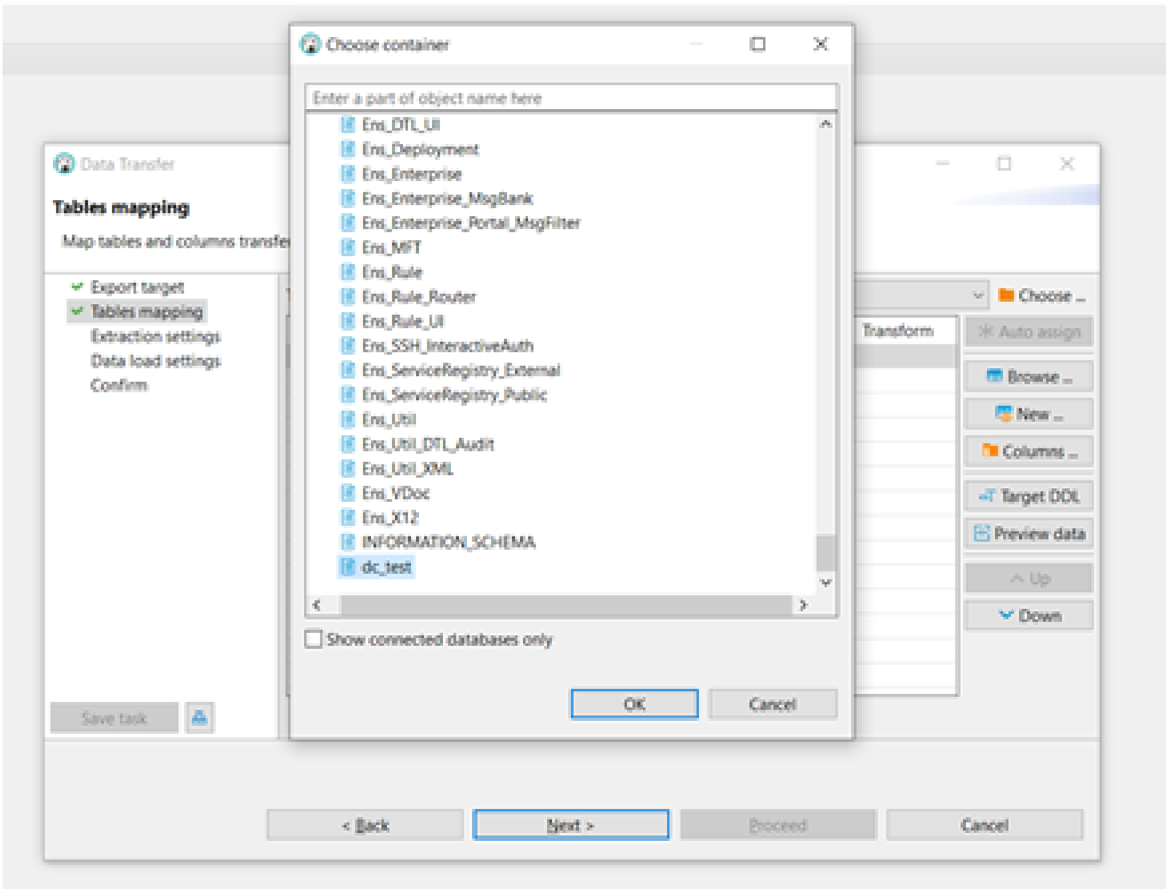


Target container: dc_test [user]

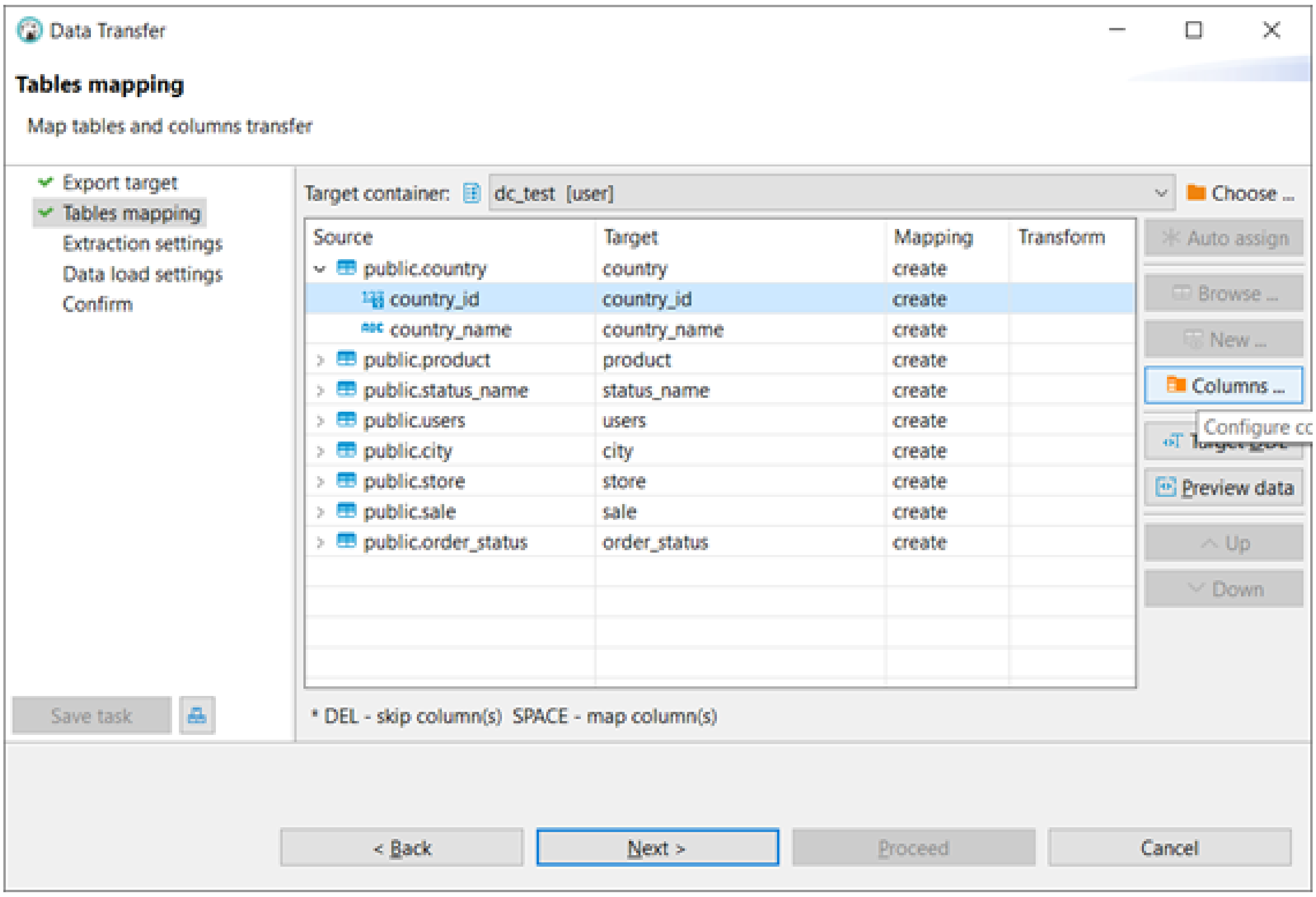
Source	Target	Mapping	Transform
public.country	country	create	
public.product	product	create	
public.status_name	status_name	create	
public.users	users	create	
public.city	city	create	
public.store	store	create	
public.sale	sale	create	
public.order_status	order_status	create	

* DEL - skip column(s) SPACE - map column(s)

4. Select dctest and click Ok.



5. Now It is necessary change some data type configurations for the target database, because the IRIS and PostgreSQL use different data types for integer and decimal values.
6. Expand public.country table, select the first field (countryid) and click Columns…



Data Transfer

Tables mapping

Map tables and columns transfer

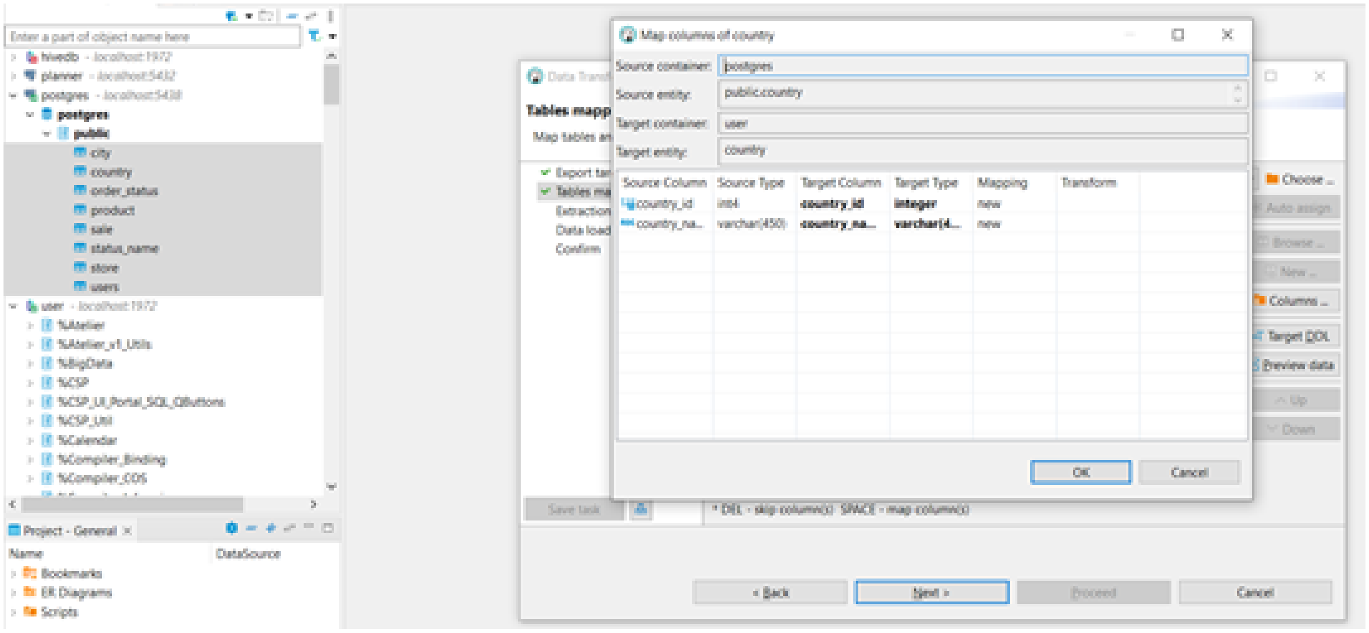
- ✓ Export target
- ✓ Tables mapping
- Extraction settings
- Data load settings
- Confirm

Target container: dc_test [user]

Source	Target	Mapping	Transform
public.country	country	create	
country_id	country_id	create	
country_name	country_name	create	
public.product	product	create	
public.status_name	status_name	create	
public.users	users	create	
public.city	city	create	
public.store	store	create	
public.sale	sale	create	
public.order_status	order_status	create	

* DEL - skip column(s) SPACE - map column(s)

7. Change the Target Type from int4 to integer and click Ok.



Map columns of country

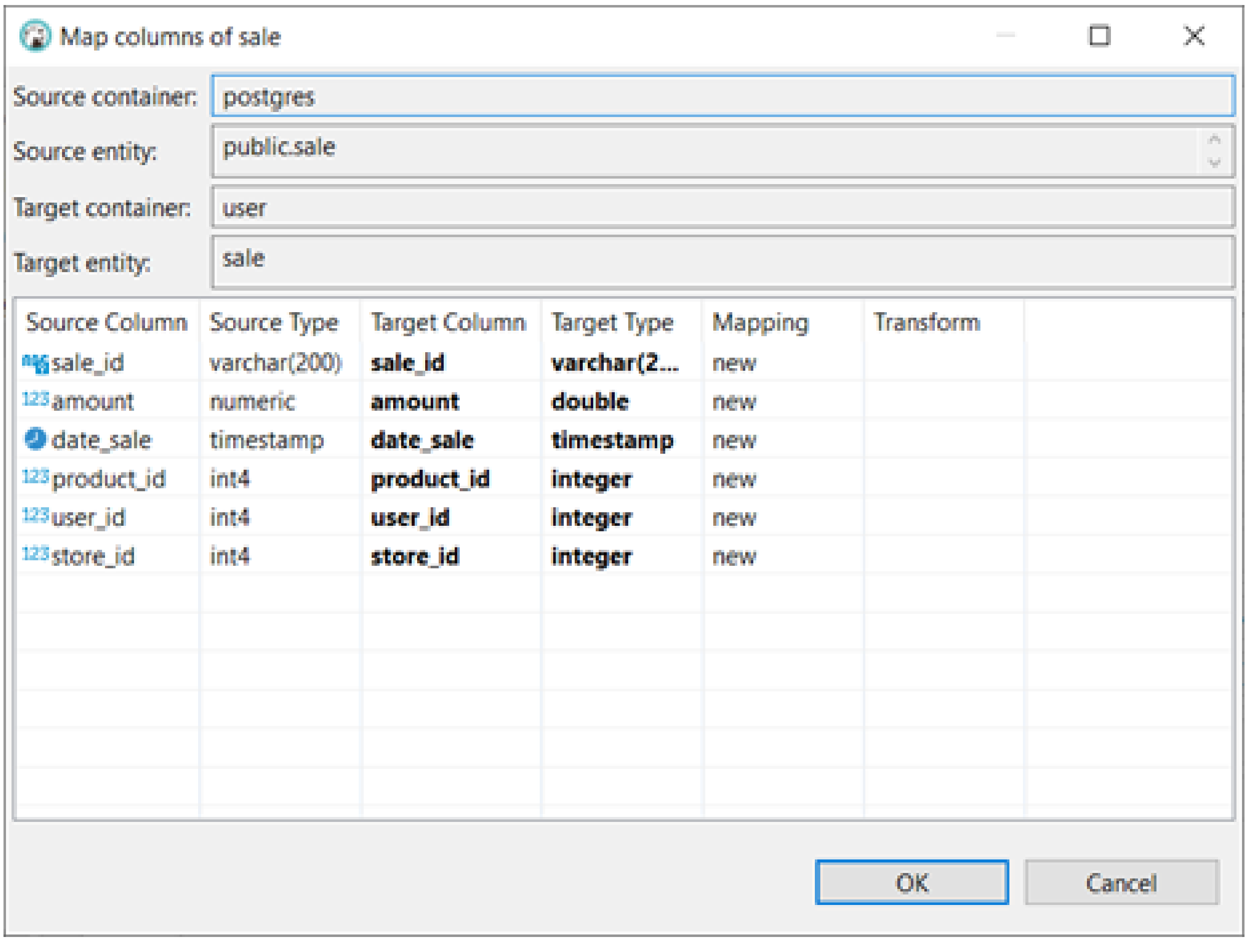
Source container: postgres
Source entity: public.country
Target container: user
Target entity: country

Source Column	Source Type	Target Column	Target Type	Mapping	Transform
country_id	int4	country_id	integer	new	
country_na...	varchar(450)	country_na...	varchar(4...	new	

8. Repeat the process to the tables

a. public.product.

b. public.statusname.
c. public.users.
d. public.city (change the type to integer for cityid and countryid).
e. public.store (change the type to integer for storeid and cityid).
f. public.sale (change the type to double for amount and integer for productid, userid and storeid)

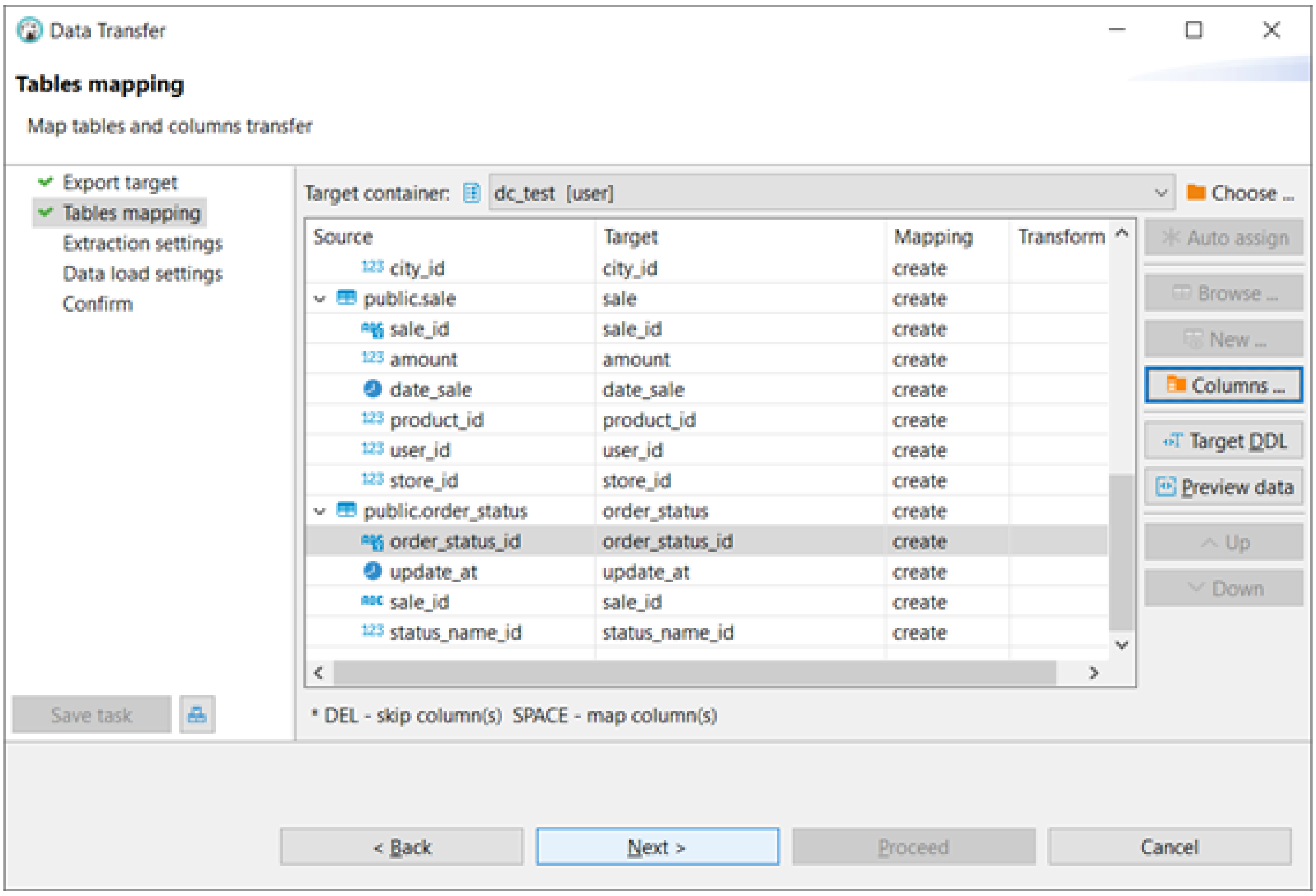

Map columns of sale

Source container:	postgres
Source entity:	public.sale
Target container:	user
Target entity:	sale

Source Column	Source Type	Target Column	Target Type	Mapping	Transform
sale_id	varchar(200)	**sale_id**	**varchar(2...**	new	
amount	numeric	**amount**	**double**	new	
date_sale	timestamp	**date_sale**	**timestamp**	new	
product_id	int4	**product_id**	**integer**	new	
user_id	int4	**user_id**	**integer**	new	
store_id	int4	**store_id**	**integer**	new	

g. public.orderstatus (change statusnameid to integer).

9. Now with the Target Data Types changed, click Next.


Data Transfer

Tables mapping

Map tables and columns transfer

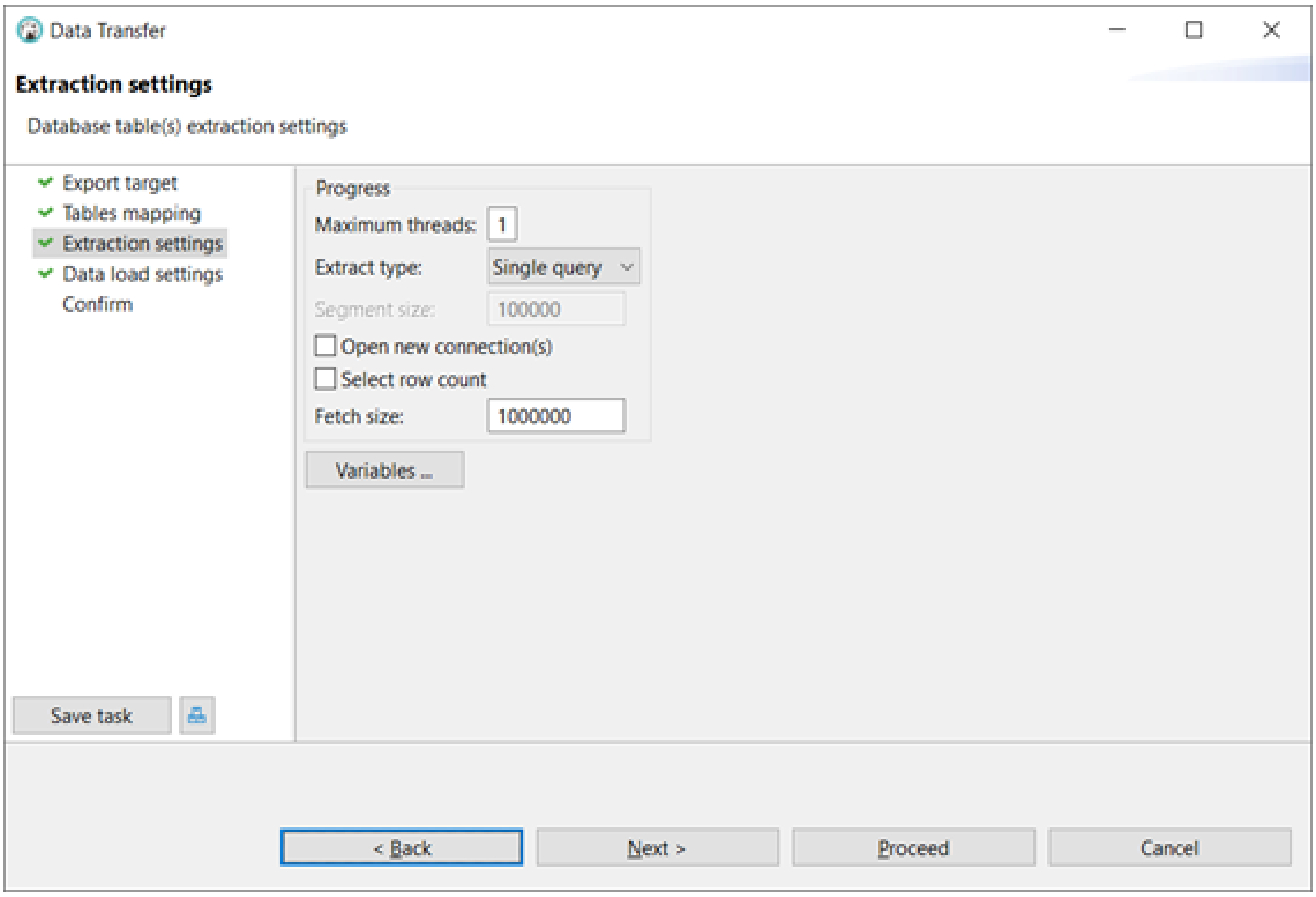
✓ Export target
✓ Tables mapping
Extraction settings
Data load settings
Confirm

Target container: dc_test [user]

Source	Target	Mapping	Transform
city_id	city_id	create	
public.sale	sale	create	
sale_id	sale_id	create	
amount	amount	create	
date_sale	date_sale	create	
product_id	product_id	create	
user_id	user_id	create	
store_id	store_id	create	
public.order_status	order_status	create	
order_status_id	order_status_id	create	
update_at	update_at	create	
sale_id	sale_id	create	
status_name_id	status_name_id	create	

* DEL - skip column(s) SPACE - map column(s)

10. Set Fetch size to 1000000 and click Next.


Data Transfer

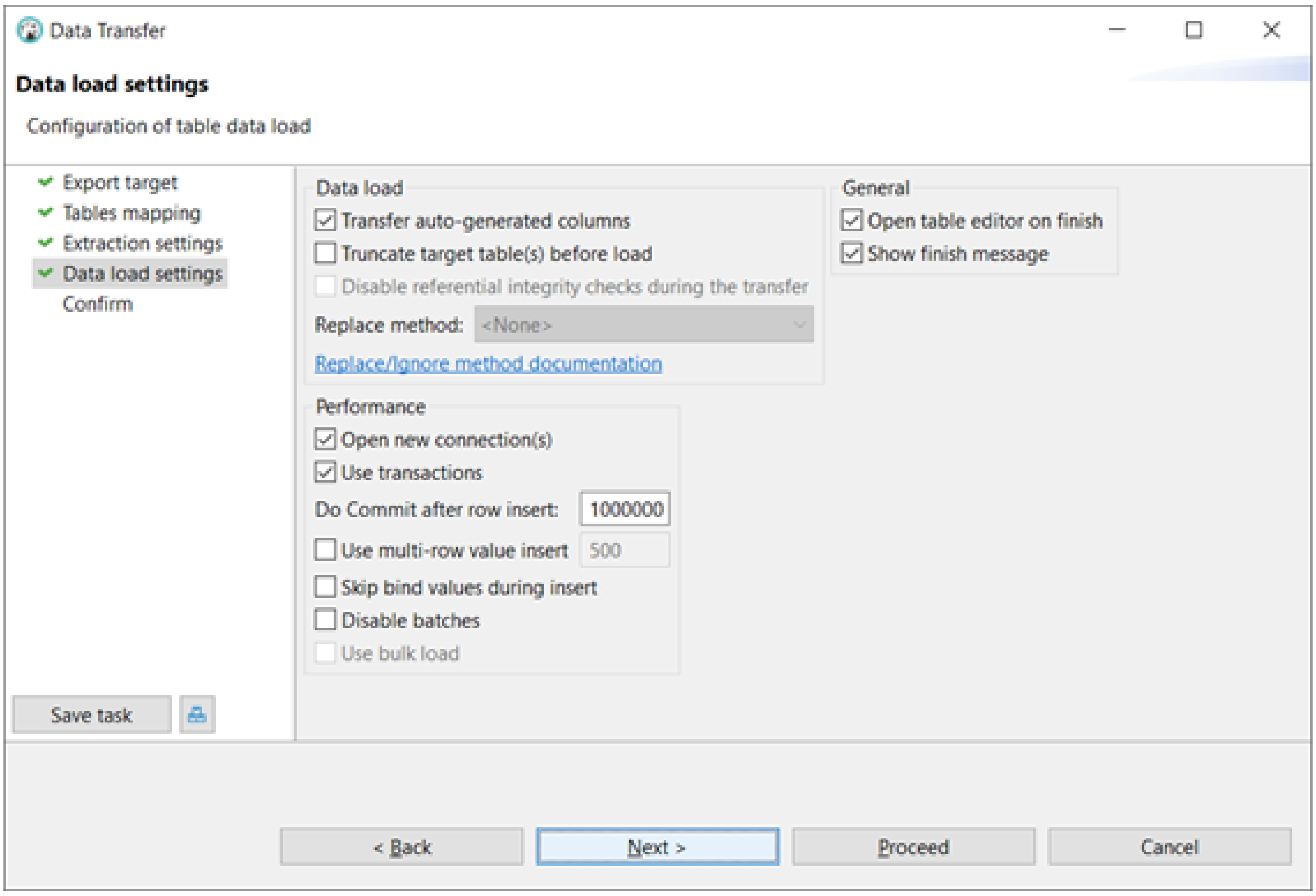
Extraction settings

Database table(s) extraction settings

✓ Export target
✓ Tables mapping
✓ Extraction settings
✓ Data load settings
Confirm

Progress
Maximum threads: 1
Extract type: Single query
Segment size: 100000
☐ Open new connection(s)
☐ Select row count
Fetch size: 1000000

11. Accept the default values to the Data load settings and click Next.



Data load settings

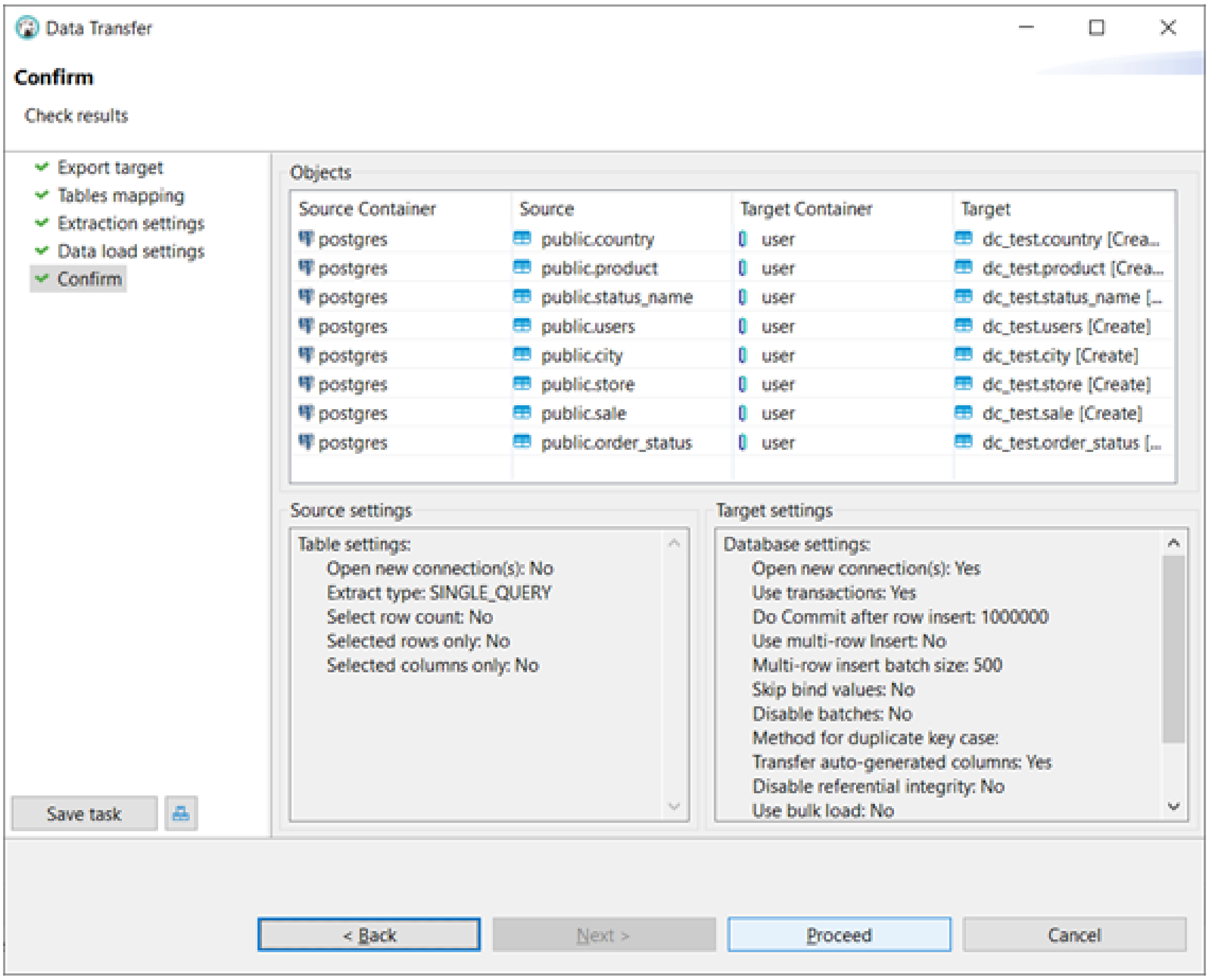
Configuration of table data load

Data load
- [✓] Transfer auto-generated columns
- [ ] Truncate target table(s) before load
- [ ] Disable referential integrity checks during the transfer
- Replace method: <None>
- Replace/Ignore method documentation

General
- [✓] Open table editor on finish
- [✓] Show finish message

Performance
- [✓] Open new connection(s)
- [✓] Use transactions
- Do Commit after row insert: 1000000
- [ ] Use multi-row value insert 500
- [ ] Skip bind values during insert
- [ ] Disable batches
- [ ] Use bulk load

12. In the Confirm click Proceed.



Confirm

Check results

- Export target
- Tables mapping
- Extraction settings
- Data load settings
- Confirm

Objects

Source Container	Source	Target Container	Target
postgres	public.country	user	dc_test.country [Crea...
postgres	public.product	user	dc_test.product [Crea...
postgres	public.status_name	user	dc_test.status_name [...
postgres	public.users	user	dc_test.users [Create]
postgres	public.city	user	dc_test.city [Create]
postgres	public.store	user	dc_test.store [Create]
postgres	public.sale	user	dc_test.sale [Create]
postgres	public.order_status	user	dc_test.order_status [...

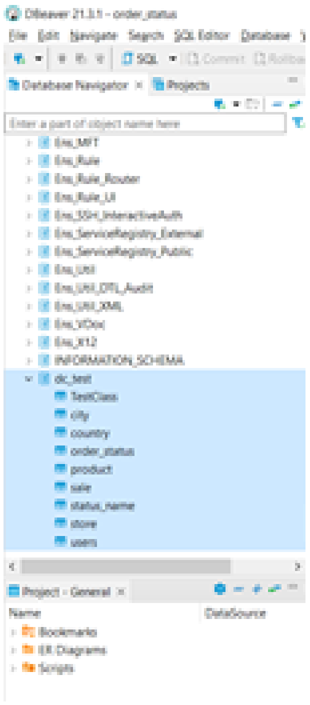
Source settings

```
Table settings:
    Open new connection(s): No
    Extract type: SINGLE_QUERY
    Select row count: No
    Selected rows only: No
    Selected columns only: No
```

Target settings

```
Database settings:
    Open new connection(s): Yes
    Use transactions: Yes
    Do Commit after row insert: 1000000
    Use multi-row Insert: No
    Multi-row insert batch size: 500
    Skip bind values: No
    Disable batches: No
    Method for duplicate key case:
    Transfer auto-generated columns: Yes
    Disable referential integrity: No
    Use bulk load: No
```

13. Now see in the Database Navigator all PostgreSQL Tables inside InterSystems IRIS dc_test schema.


DBeaver 21.3.1 - order_status
File Edit Navigate Search SQL Editor Database
SQL Commit Rollback
Database Navigator
Projects
Enter a part of object name here

```
> Ens_MFT
> Ens_Rule
> Ens_Rule_Router
> Ens_Rule_UI
> Ens_SSH_InteractiveAuth
> Ens_ServiceRegistry_External
> Ens_ServiceRegistry_Public
> Ens_Util
> Ens_Util_DTL_Audit
> Ens_Util_XML
> Ens_VDoc
> Ens_X12
> INFORMATION_SCHEMA
v dc_test
    TestClass
    city
    country
    order_status
    product
    sale
    status_name
    store
    users
```

Project - General
Name
DataSource
Bookmarks
ER Diagrams
Scripts

The migration process was very simple for tables, but for views, functions, triggers and stored procedures, you need to rewrite the SQL source code using ObjectScript or SQL.

What do you get by migrating to IRIS?

This list summarizes what you get in IRIS:

- API Management.
- Visual Reports (IRIS Reports).
- AutoML (IntegratedML).
- Multilanguage application/data development (Python, Java, .NET, JavaScript).
- ESB.
- BI/Analytics.
- NLP.
- Microservices development.
- Multimodel database (SQL, JSON, Analytical Cubes, Object Oriented).
- Sharding.

In summary, when migrating to IRIS you get a data platform, when before you only had a database.

#Data Import and Export #InterSystems IRIS