

Table of Contents

DATA MANAGEMENT TOOLS	4
IMPORT WIZARD	6
<i>Setting Import File Format</i>	7
<i>Setting Source File Paths and Encoding</i>	8
Importing ODBC	9
<i>Setting Format Options</i>	12
<i>Setting Additional Options</i>	15
<i>Setting Target Tables</i>	16
<i>Adjusting Field Structures and Mapping Fields</i>	18
<i>Selecting Import Mode</i>	20
<i>Saving and Confirming Import</i>	22
EXPORT WIZARD	23
<i>Setting Export File Format</i>	24
<i>Setting Destination File Paths and Encoding</i>	25
<i>Selecting Table Columns for Export</i>	26
<i>Setting Additional Options</i>	27
<i>Saving and Confirming Export</i>	29
DATA TRANSFER (AVAILABLE ONLY IN FULL VERSION)	30
<i>General Settings for Data Transfer</i>	32
<i>Advanced Settings for Same Server Type Data Transfer</i>	33
<i>Advanced Settings for Cross Server Data Transfer (Available only in Navicat Premium)</i>	37
Advanced Settings for Transferring from MySQL to Oracle	38
Advanced Settings for Transferring from MySQL to PostgreSQL	40
Advanced Settings for Transferring from MySQL to SQLite	42
Advanced Settings for Transferring from MySQL to SQL Server	44
Advanced Settings for Transferring from Oracle to MySQL	46
Advanced Settings for Transferring from Oracle to PostgreSQL	48
Advanced Settings for Transferring from Oracle to SQLite	50
Advanced Settings for Transferring from Oracle to SQL Server	52
Advanced Settings for Transferring from PostgreSQL to MySQL	54
Advanced Settings for Transferring from PostgreSQL to Oracle	56
Advanced Settings for Transferring from PostgreSQL to SQLite	58
Advanced Settings for Transferring from PostgreSQL to SQL Server	60
Advanced Settings for Transferring from SQLite to MySQL	62
Advanced Settings for Transferring from SQLite to Oracle	64
Advanced Settings for Transferring from SQLite to PostgreSQL	66
Advanced Settings for Transferring from SQLite to SQL Server	68

Advanced Settings for Transferring from SQL Server to MySQL	70
Advanced Settings for Transferring from SQL Server to Oracle	72
Advanced Settings for Transferring from SQL Server to PostgreSQL	74
Advanced Settings for Transferring from SQL Server to SQLite	76
<i>Data Transfer Message Log</i>	78
DATA SYNCHRONIZATION (AVAILABLE ONLY IN FULL VERSION)	79
<i>General Settings for Data Synchronization</i>	81
<i>Advanced Settings for Data Synchronization</i>	82
<i>Data Synchronization Result</i>	83
STRUCTURE SYNCHRONIZATION (AVAILABLE ONLY IN FULL VERSION & ONLY FOR MYSQL, ORACLE, POSTGRESQL AND SQL SERVER)	84
<i>General Settings for MySQL Structure Synchronization</i>	86
<i>General Settings for Oracle Structure Synchronization</i>	88
<i>General Settings for PostgreSQL Structure Synchronization</i>	90
<i>General Settings for SQL Server Structure Synchronization</i>	92
<i>Structure Synchronization Result</i>	94
<i>Structure Synchronization Message Log</i>	96
BACKUP/RESTORE (AVAILABLE ONLY IN FULL VERSION & ONLY FOR MYSQL, POSTGRESQL AND SQLITE)	97
<i>Backup</i>	100
General Settings for Backup	101
Object Selection for Backup	102
Advanced Settings for Backup	103
Backup Message Log	104
<i>Restore</i>	105
General Settings for Restore	106
Restore Message Log	107
<i>Extract SQL</i>	108
BATCH JOB/SCHEDULE (AVAILABLE ONLY IN FULL VERSION)	109
<i>General Settings for Batch Job/Schedule</i>	112
<i>Advanced Settings for Batch Job/Schedule</i>	114
Setup Schedule	115
<i>Batch Job/Schedule Message Log</i>	118
<i>Batch Job Converter (Available only for Navicat Premium)</i>	119
Selecting Batch Jobs	120
Setting Convert Options	121
Starting Convert	122
CONSOLE	123
<i>MySQL Console</i>	124

Example of Using MySQL Console	125
<i>Oracle Console</i>	126
Example of using SQL*Plus	127
<i>PostgreSQL Console</i>	128
Example of Using PostgreSQL Console	129
<i>SQLite Console</i>	130
Example of Using SQLite Console	131
<i>SQL Server Console</i>	132
Example of Using SQL Server Console	133
DUMP SQL FILE	134
EXECUTE SQL FILE	135
CONVERT TO SQLITE 3 (AVAILABLE ONLY FOR SQLITE)	136
VIEW DATABASE/SCHEMA/TABLE STRUCTURE (AVAILABLE ONLY IN FULL VERSION)	137
LOG FILES	138

Data Management Tools

Navicat provides a number of powerful tools for working with the databases.

The following tools are available:

[Import Wizard](#)

Imports data from TXT, CSV, XML, DBF, MS Excel and ODBC.

[Export Wizard](#)

Exports data to TXT, CSV, DBF, XML and MS Excel.

[Data Transfer](#)

Transfers tables/views/procedures/functions/sequences/events between databases/schemas or plain text file.

[Data Synchronization](#)

Synchronizes data in different databases/schemas to be kept up-to-date so that each repository contains the same information.

[Structure Synchronization](#)

Compares the structure of two similar databases/schemas and produces a set of alter statements in MySQL, Oracle, PostgreSQL and SQL Server.

[Backup/Restore](#)

Allows you to backup/restore your databases/schemas for MySQL, PostgreSQL and SQLite.

[Batch Job/Schedule](#)

Allows you to schedule a batch job which being executed at a specified time and support e-mail notification service.

[Console](#)

Provides interactive text-based screen for user query input and result output from MySQL, Oracle, PostgreSQL, SQLite and SQL Server.

[Dump SQL File](#)

Dumps database/schema/table(s) to SQL file.

[Execute SQL File](#)

Executes SQL file.

[Convert to SQLite 3](#)

Converts a SQLite 2 file into a SQLite 3 file.

[Print Structure](#)

Prints database/schema/table structure.

[Log Files](#)

Keeps track on the actions (e.g. SQL statements being executed) have been performed in Navicat.

Import Wizard

Import Wizard allows you to import data to a table from CSV, TXT, XML, DBF, MS Excel and ODBC. You can save your settings as a profile for setting schedule.

Note: Navicat Essentials version supports to import text-based files, such as TXT, CSV and XML file.

Note: For importing ODBC, you need to install the corresponding driver.

Note: You can drag a supported file to the table pane or a database in the connection tree. Navicat will popup the import wizard. (If existing table is highlighted, Navicat will import the file to the highlighted table, otherwise, import the file to a new table.)

To open the Import Wizard, click **Import** from the table object pane toolbar.

- [Setting Import File Format](#)
- [Setting Source File Paths and Encoding](#)
- [Setting Format Options](#)
- [Setting Additional Options](#)
- [Setting Target Tables](#)
- [Adjusting Field Structures and Mapping Fields](#)
- [Selecting Import Mode](#)
- [Saving and Confirming Import](#)

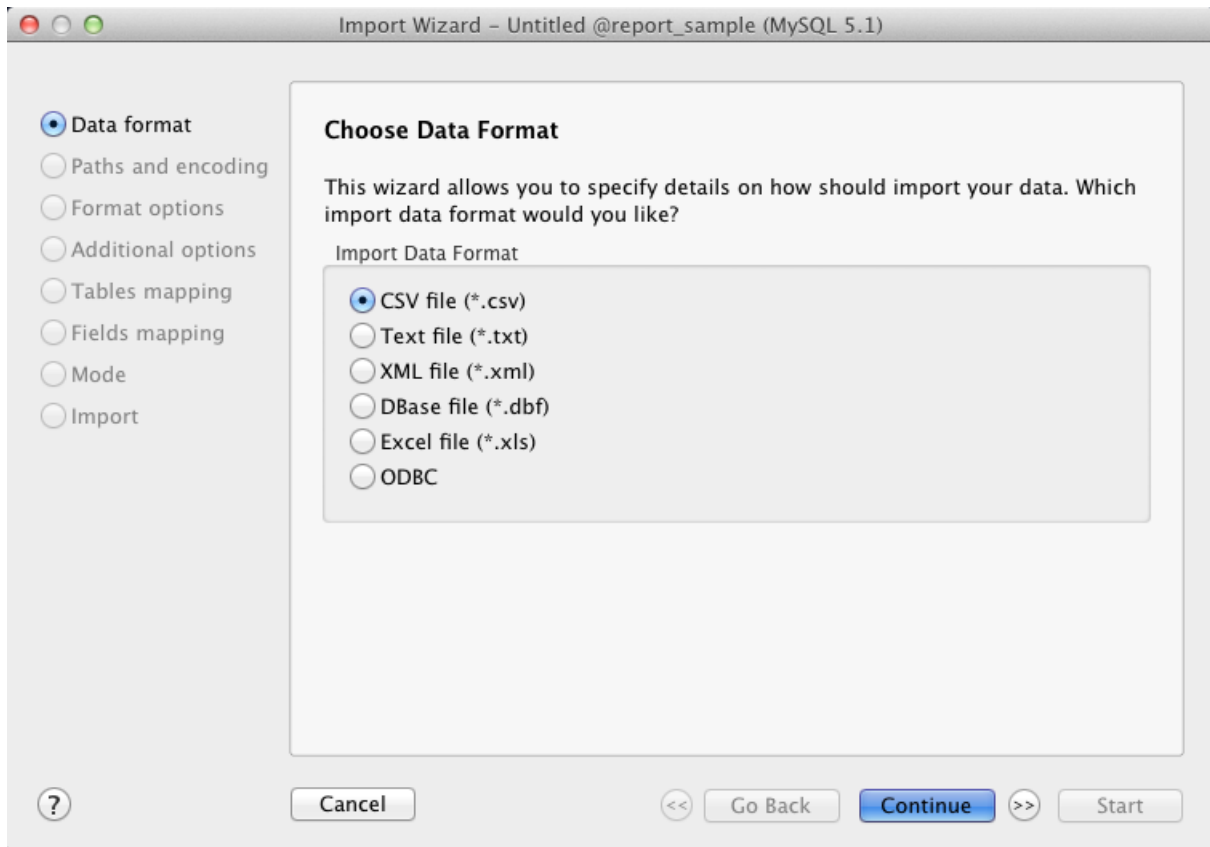
To run a saved import profile from the command line

- Create and save the import profile.
- In terminal, type the command (see Command for details)

Setting Import File Format

Select one of the available import types for the source file.

Note: Navicat Essentials only supports importing from TXT, CSV and XML file.



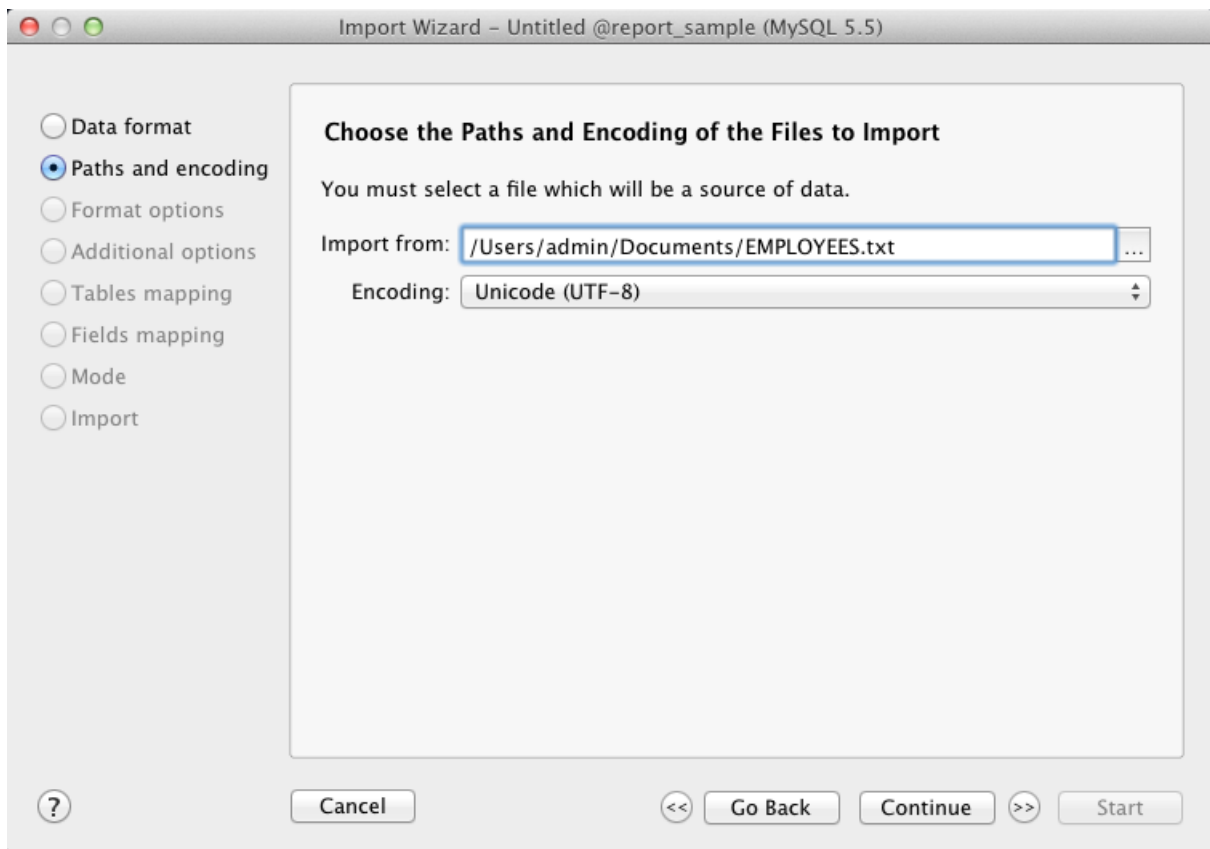
Setting Source File Paths and Encoding

Import from

Browser the source file name.

Encoding

Select the encoding for the source file.

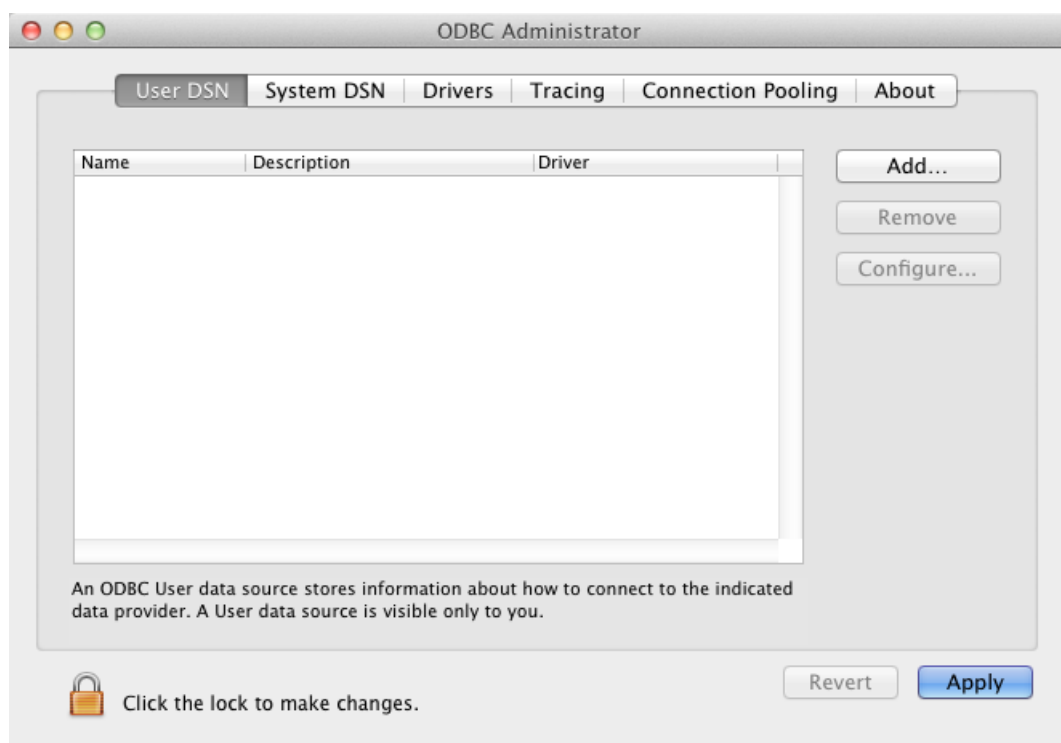


Importing ODBC

Setting Up an ODBC Data Source Connection

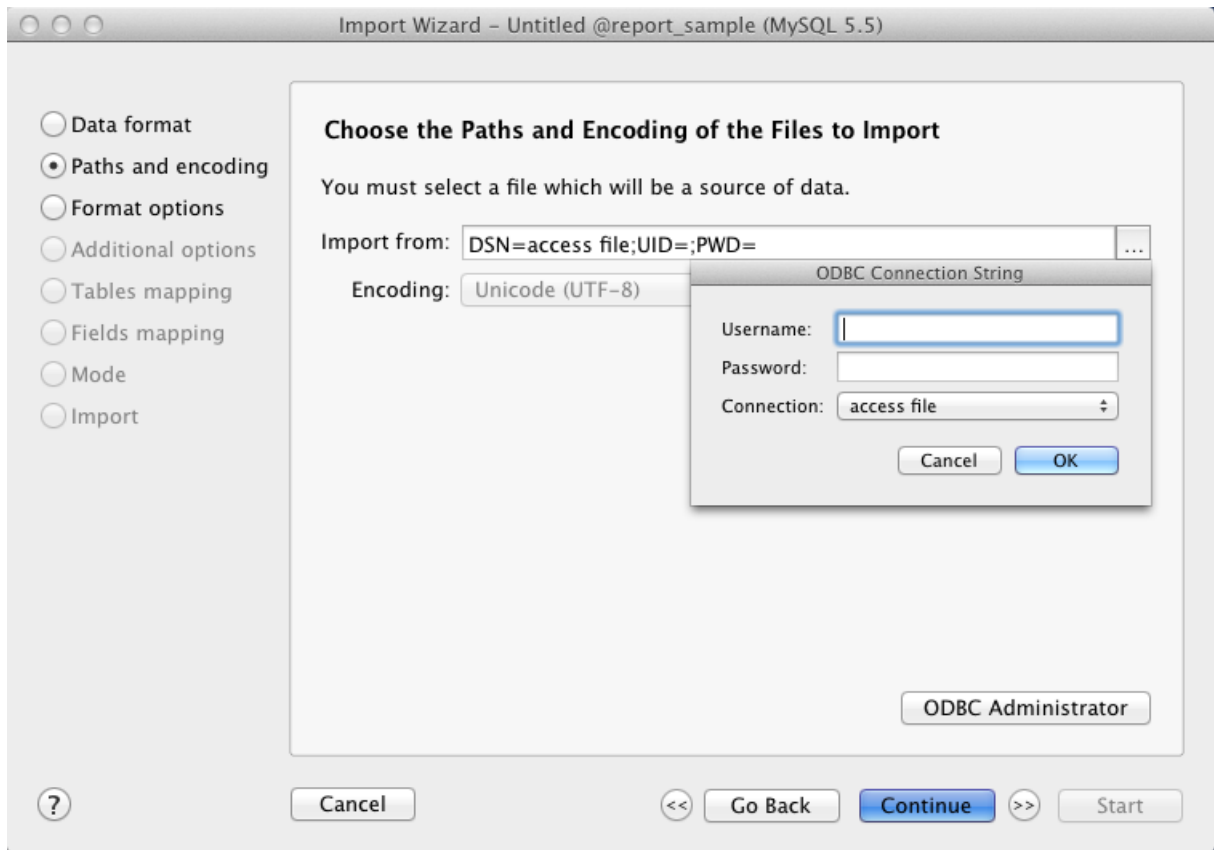
1. To setup the connection of the data source, you need to install the corresponding driver.
2. Then, setup the DSN (Data Source Name) using the **ODBC Administrator**.

Note: You can consult with the driver provider about how to setup the DSN.

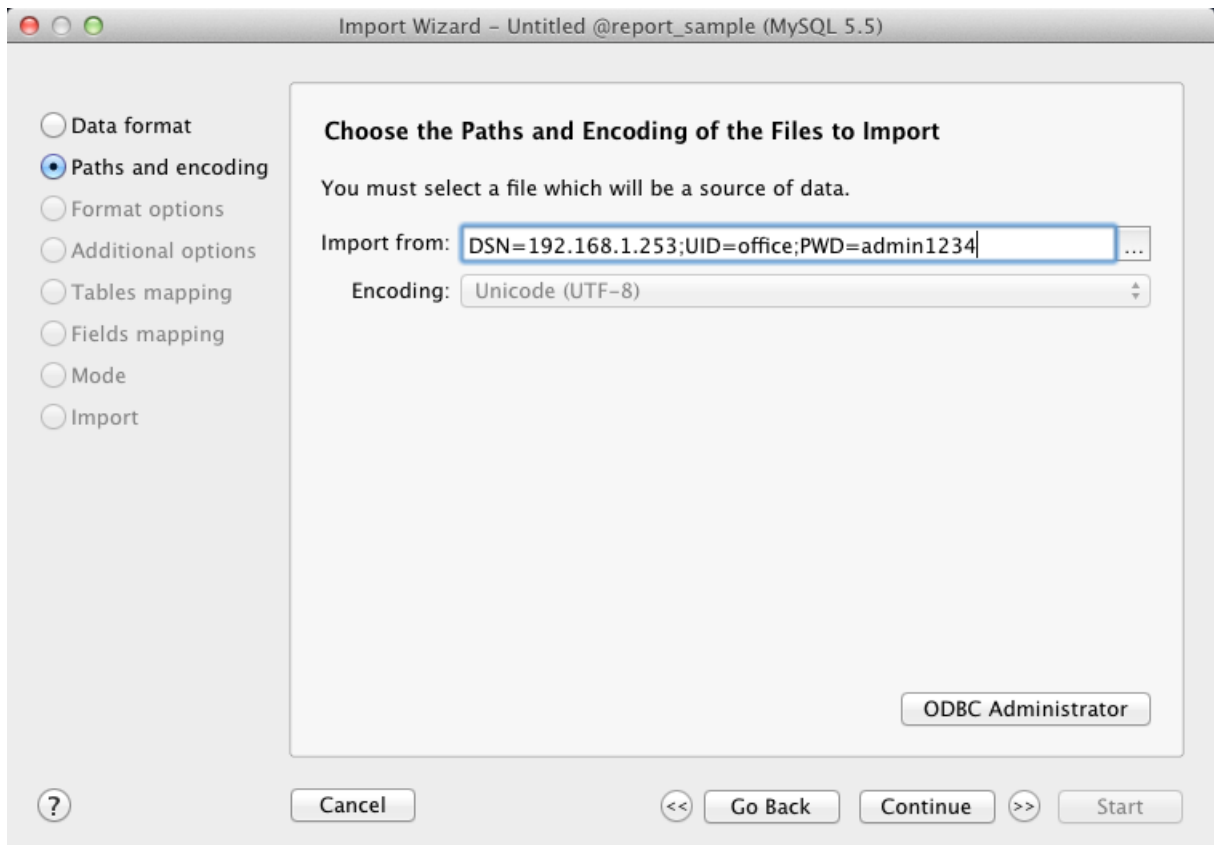


Connecting to ODBC data source in Navicat

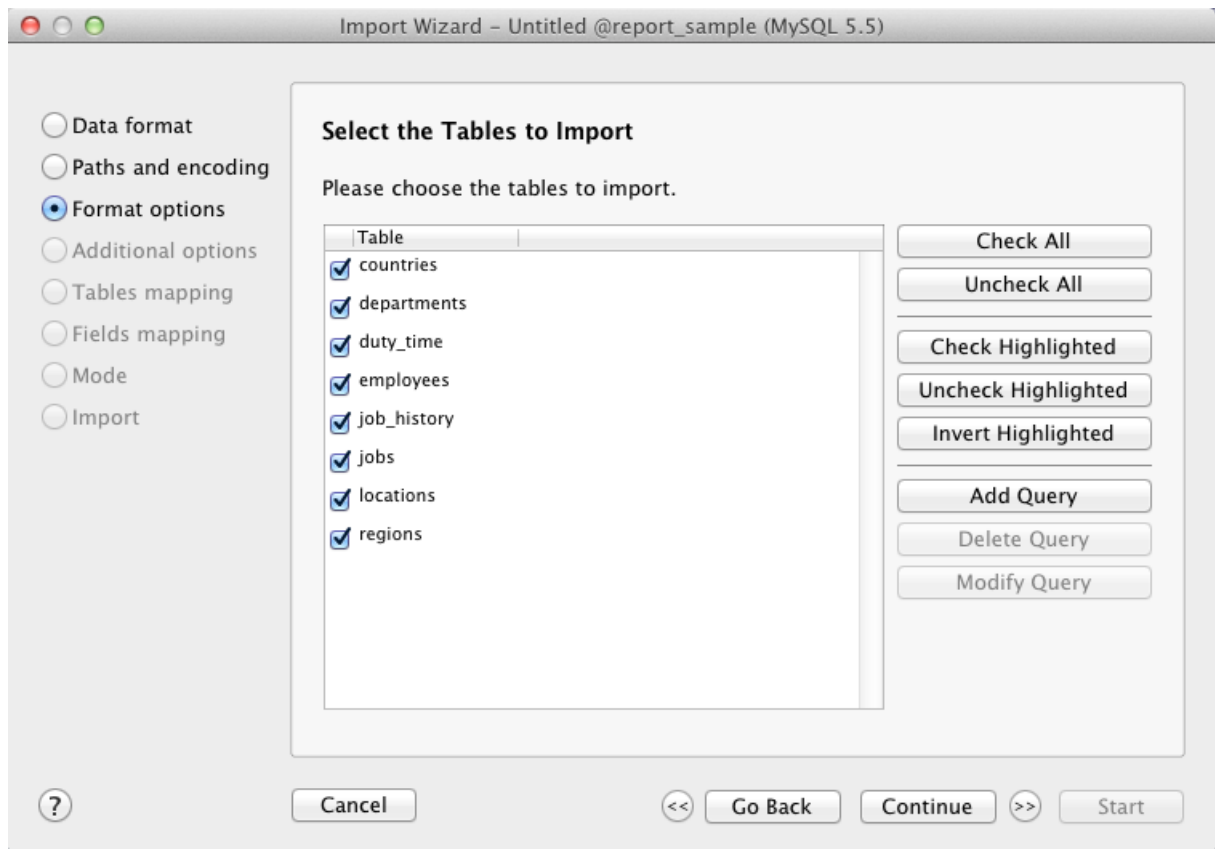
1. Click the **Import from** button in Paths and Encoding of the Import Wizard. Choose the data source from the **Connection** drop-down list and provide valid **Username** and **Password**.



Or, type the connection string directly in the **Import from** text box.



2. All available tables will be included in the list if connection success. Just simply choose the tables you wish to import or specify a query using **Add Query** button.

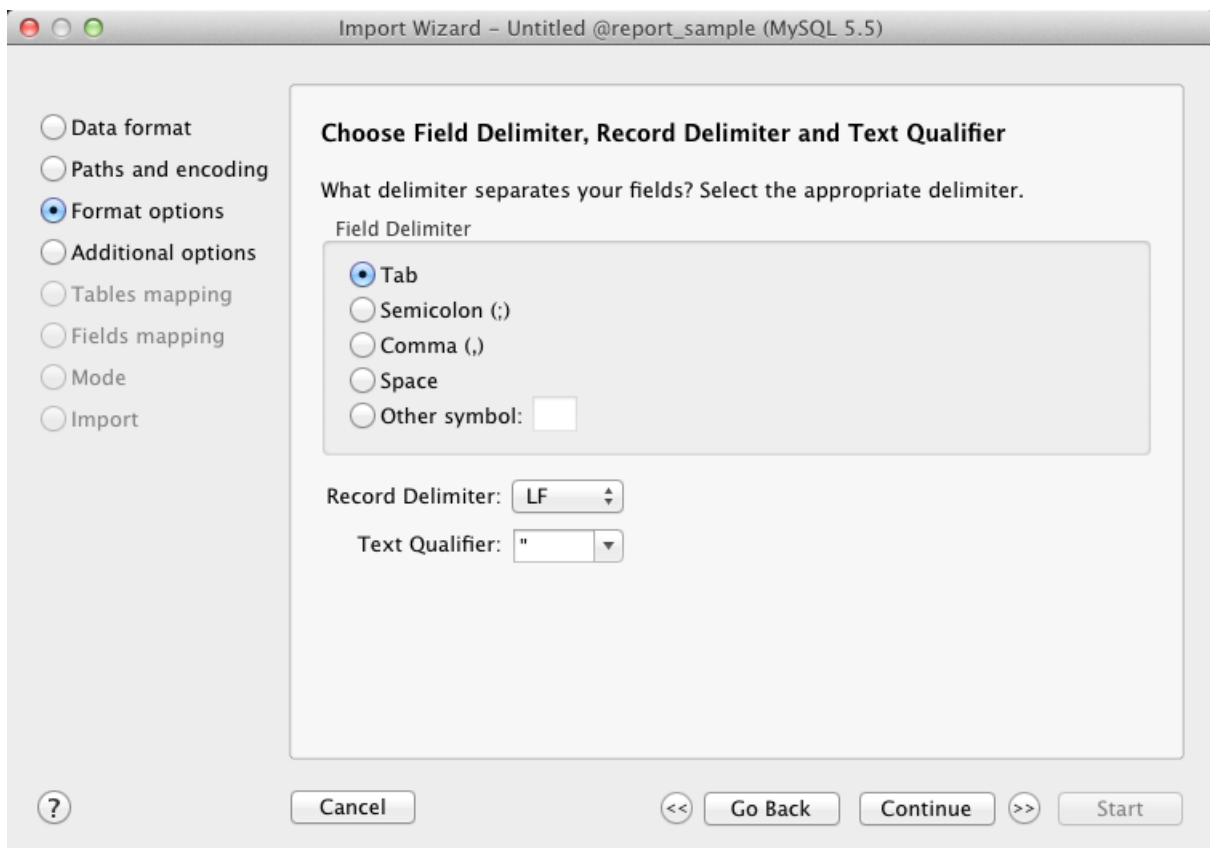


Setting Format Options

Format options specifies for file type.

TXT, CSV

Define **Field Delimiter**, **Record Delimiter** and **Text Qualifier** for file. Record delimiter indicates how the file recognizes as new record (row).



XML

Define tag to identify table row.

Consider tag's attributes as table's field

For example:

```
<row age="17">
<id>1</id>
<name>sze</name>
</row>
```

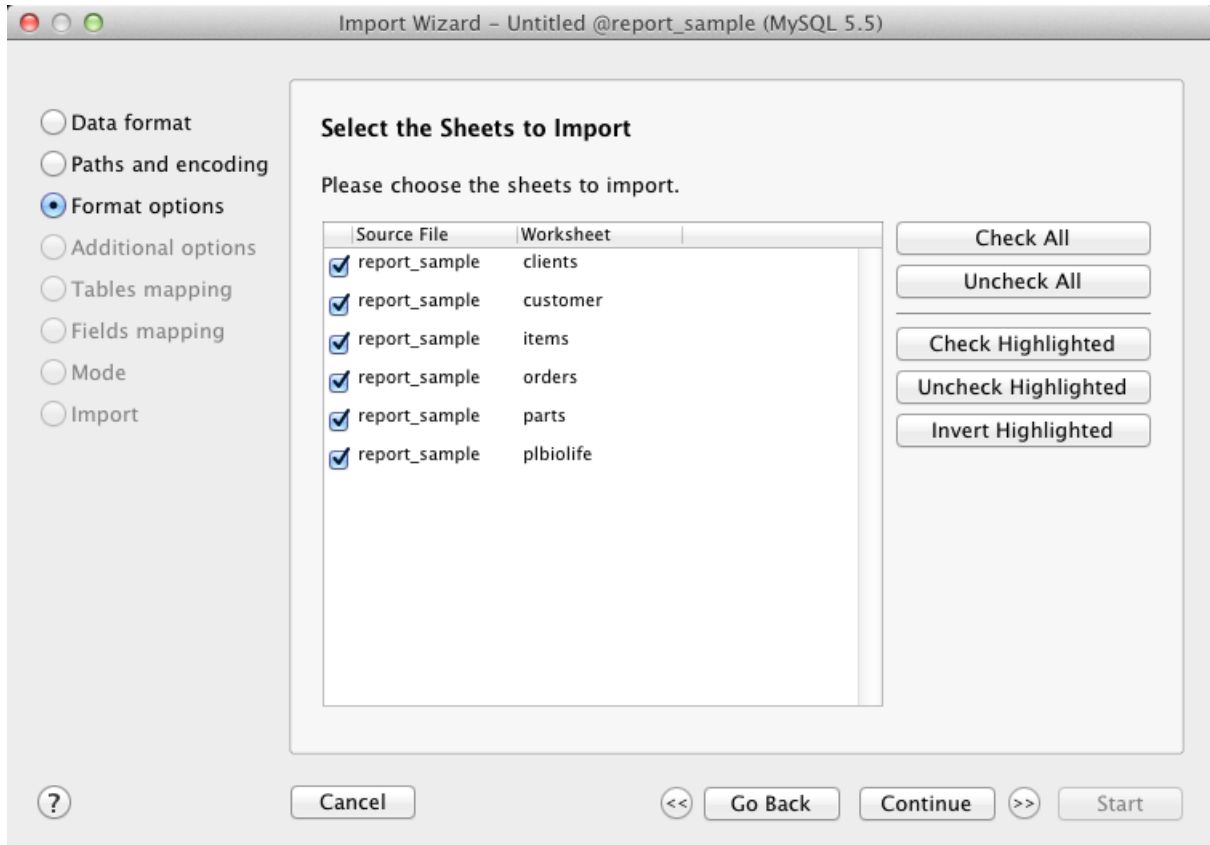
With this option is on, Navicat will recognizes "age" as a table field together with "id" and "name", otherwise, only "id" and "name" will be imported as table fields.

Note: Navicat does not support multiple level of XML file.



Excel

Select the worksheet(s) you wish to import from the excel source file. Each worksheet will be imported as a separate table.



Setting Additional Options

Import Wizard provides a number of options for setting common format.

First Row

First row indicates which row should Navicat start reading the actual data.

Last Row

Last row indicates which row should Navicat stop reading the actual data.

Field Name Row

Field name row indicates which row should Navicat recognize as Column Title.

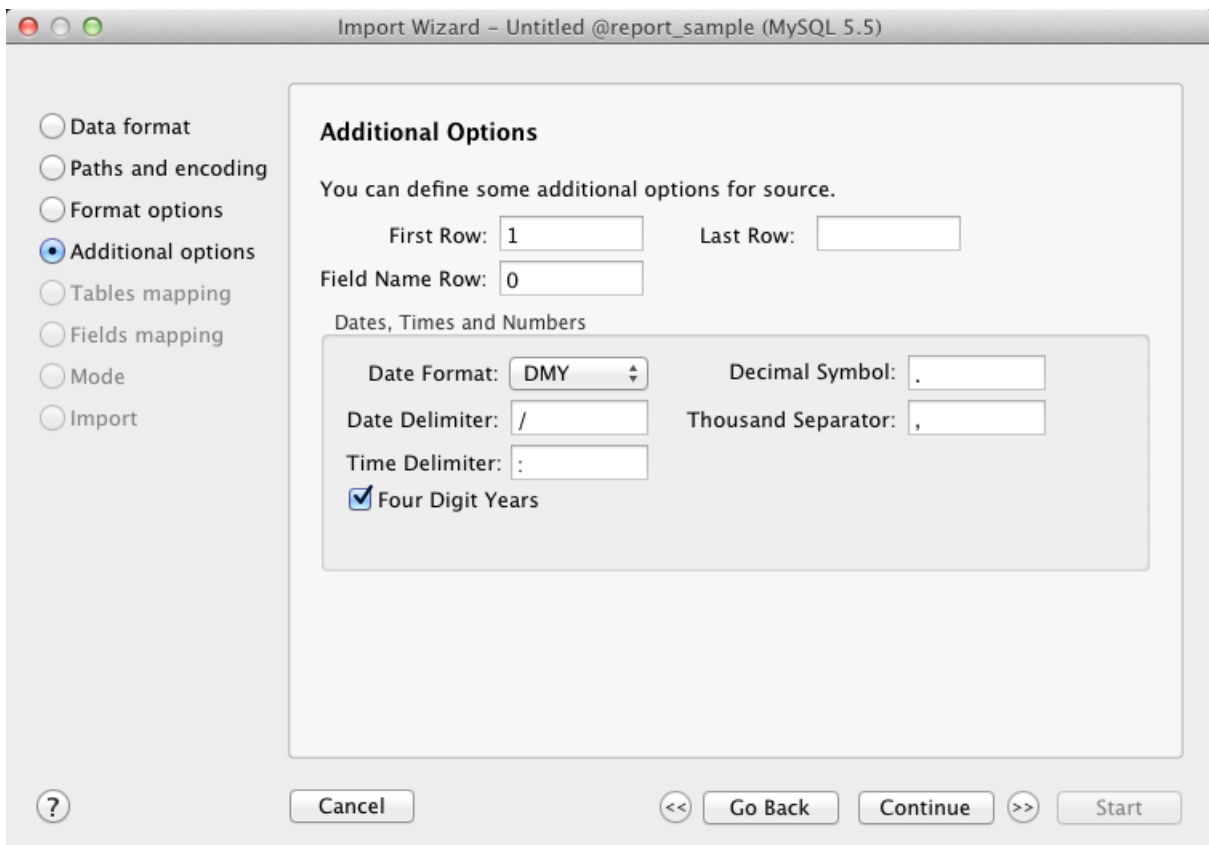
Note: If no column title is defined for the file, please enter **1** for First row and **0** for Field name row.

Dates, Times and Numbers

Define the formats of the date, time and number.

Four Digit Years

Check this option to display four digits for years.



The screenshot shows the 'Additional Options' dialog box in the Navicat Import Wizard. The window title is 'Import Wizard - Untitled @report_sample (MySQL 5.5)'. On the left, there is a sidebar with radio buttons for 'Data format', 'Paths and encoding', 'Format options', 'Additional options' (selected), 'Tables mapping', 'Fields mapping', 'Mode', and 'Import'. The main area is titled 'Additional Options' and contains the following fields and options:

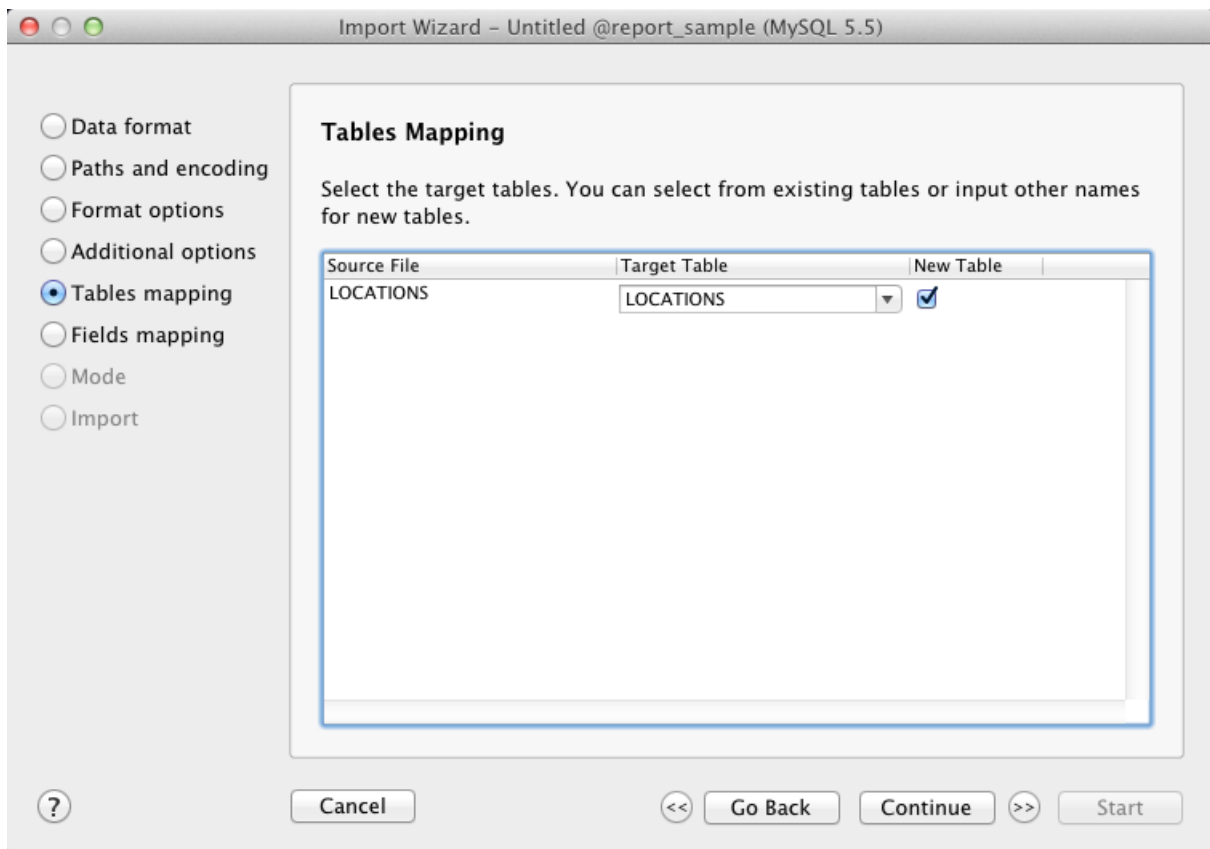
- 'You can define some additional options for source.'
- 'First Row: 1' (text input)
- 'Last Row: ' (text input)
- 'Field Name Row: 0' (text input)
- 'Dates, Times and Numbers' section:
 - 'Date Format: DMY' (dropdown menu)
 - 'Decimal Symbol: .' (text input)
 - 'Date Delimiter: /' (text input)
 - 'Thousand Separator: ,' (text input)
 - 'Time Delimiter: :' (text input)
 - Four Digit Years

At the bottom, there are buttons for '?', 'Cancel', '<< Go Back', 'Continue >>', and 'Start'.

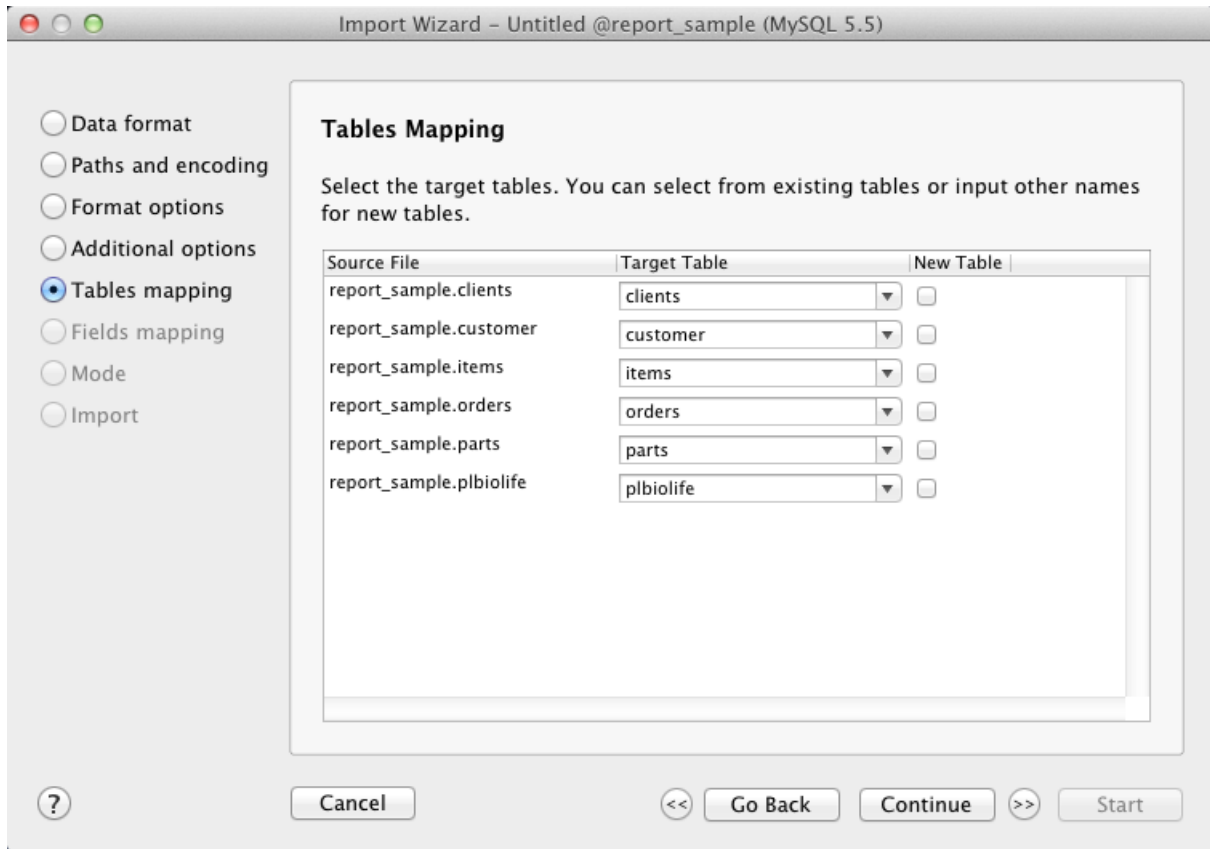
Setting Target Tables

You are allowed to define a new table name or choose to import into the existing table from the drop-down list.

Note: If you type a new table name in **Target Table**, the box in **New Table** will be checked automatically.



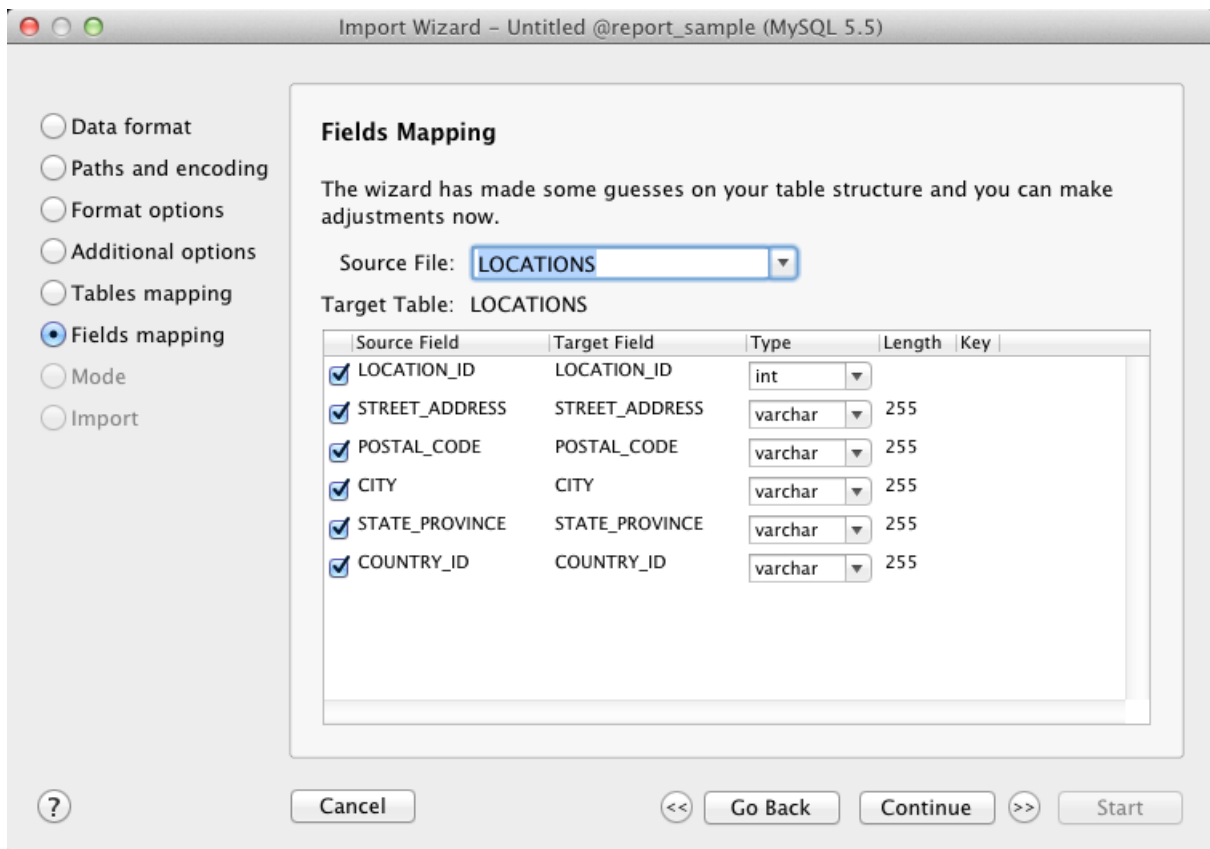
For importing multiple tables, all tables will be shown in the list.



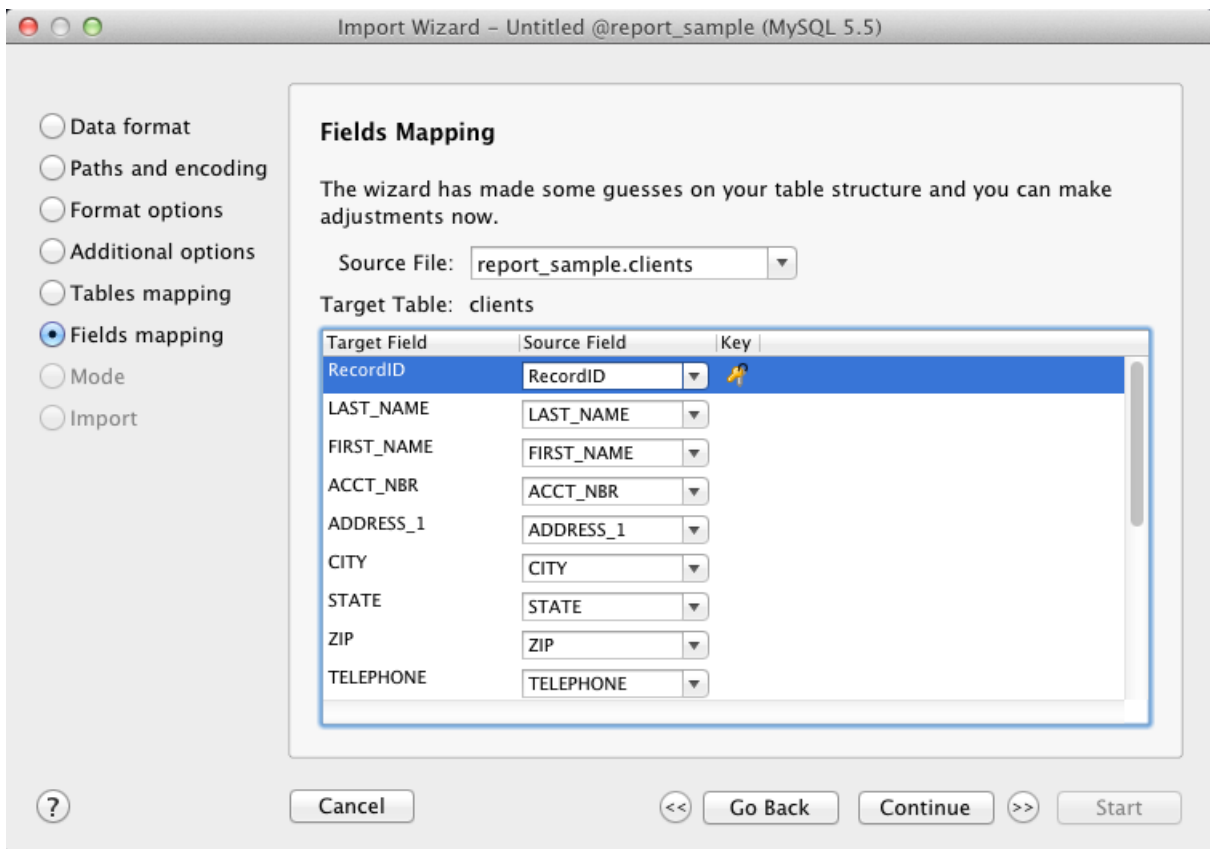
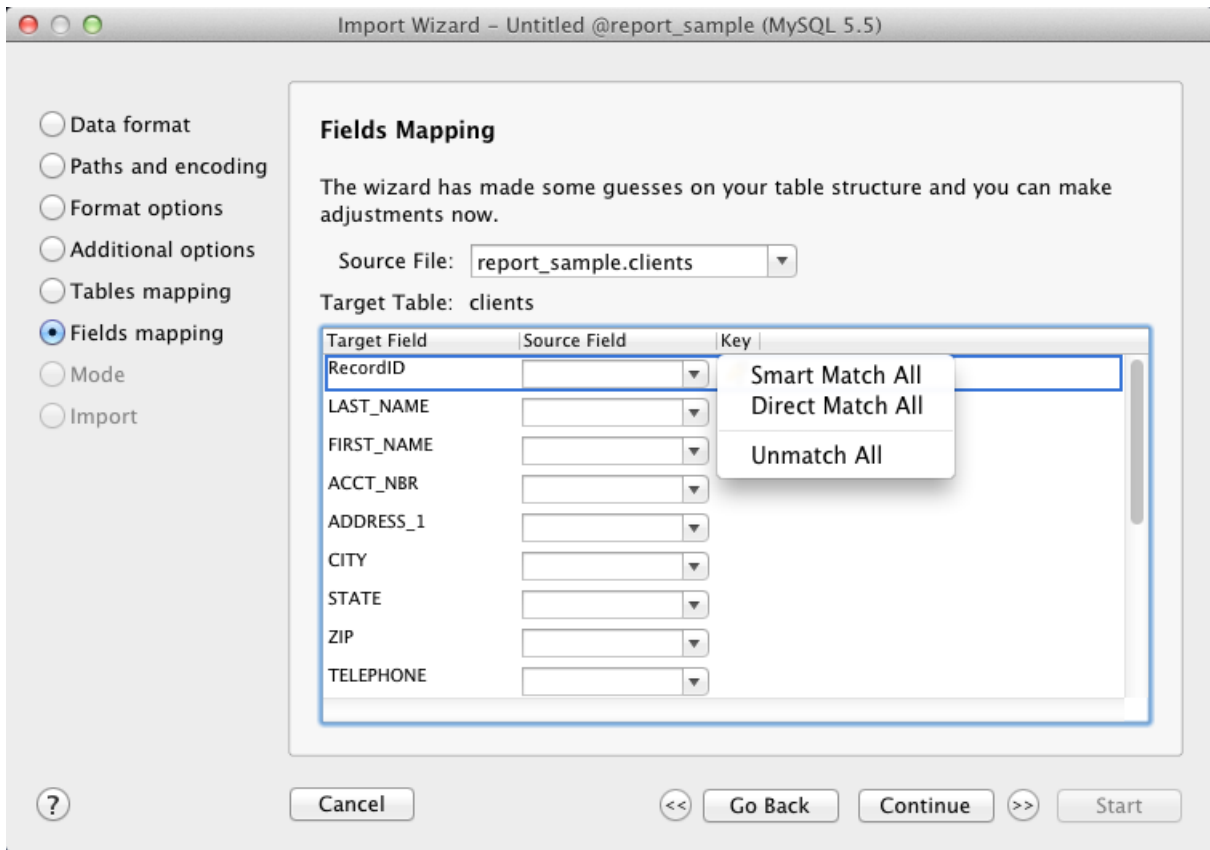
Adjusting Field Structures and Mapping Fields

Navicat will make assumption on the field types and length in the source table. You are allowed to choose your desired type from the drop-down list.

Hint: For importing multiple tables, select the other tables from the **Source File** drop-down list.

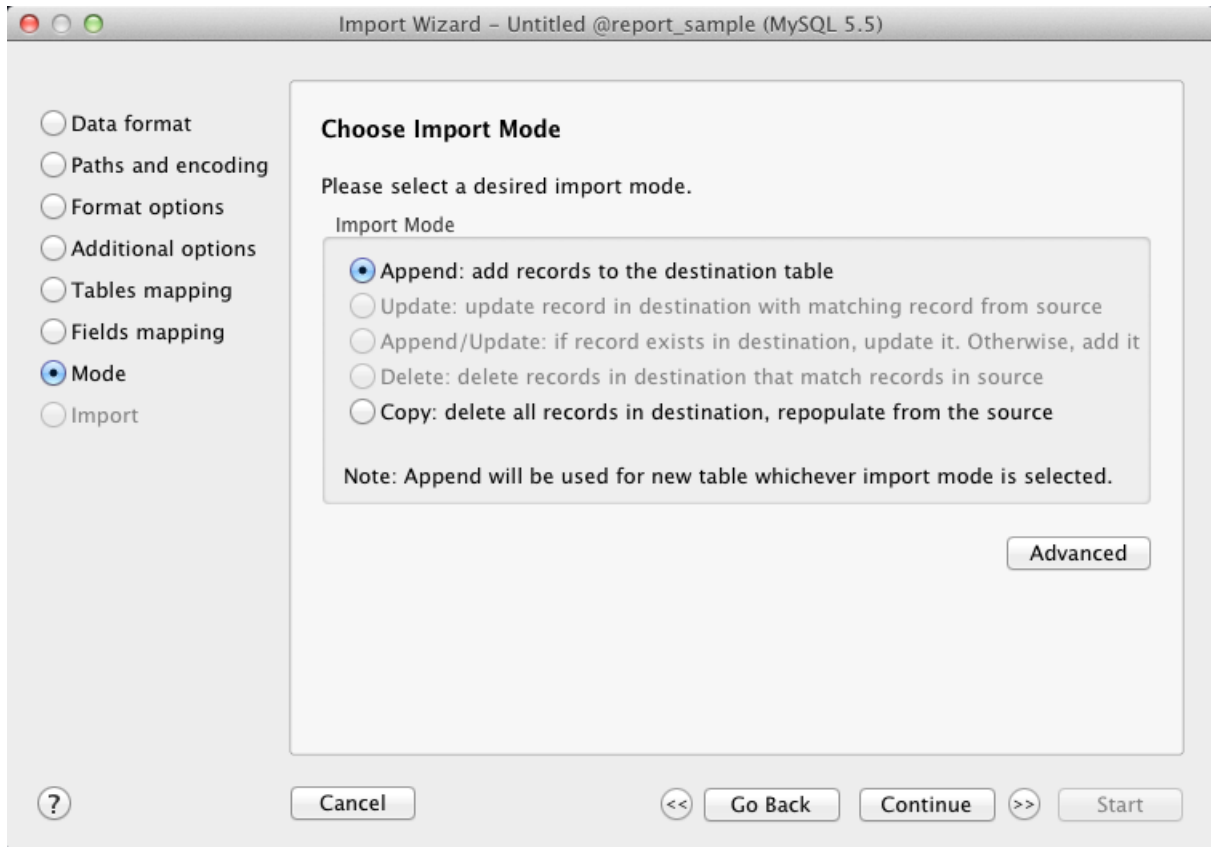


If you are importing your data into the existing table, then you might need to map the source field names manually to the destination table or just control-click and select **Smart Match All**, **Direct Match All** and **Unmatch All** from the popup menu for quick mapping.





Selecting Import Mode

Select the import mode that define how the data being imported.



Hint: To activate the remaining options, you must enable Primary Key in step Adjusting Field Structures and Mapping Fields.

Target Field	Source Field	Key
PartNo	PartNo	
VendorNo	VendorNo	
Description	Description	
OnHand	OnHand	
OnOrder	OnOrder	
Cost	Cost	
ListPrice	ListPrice	

Advanced

Use extended insert statement (Available only for MySQL)

Inserts records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

Run multiple insert statements (Available only for PostgreSQL and SQL Server)

Checks this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use empty string as NULL

Imports **NULL** value if the source data field contains empty string.

Use Foreign Key constraint (Available only for MySQL)

Adds foreign key if there is foreign key relations between tables.

Continue on error

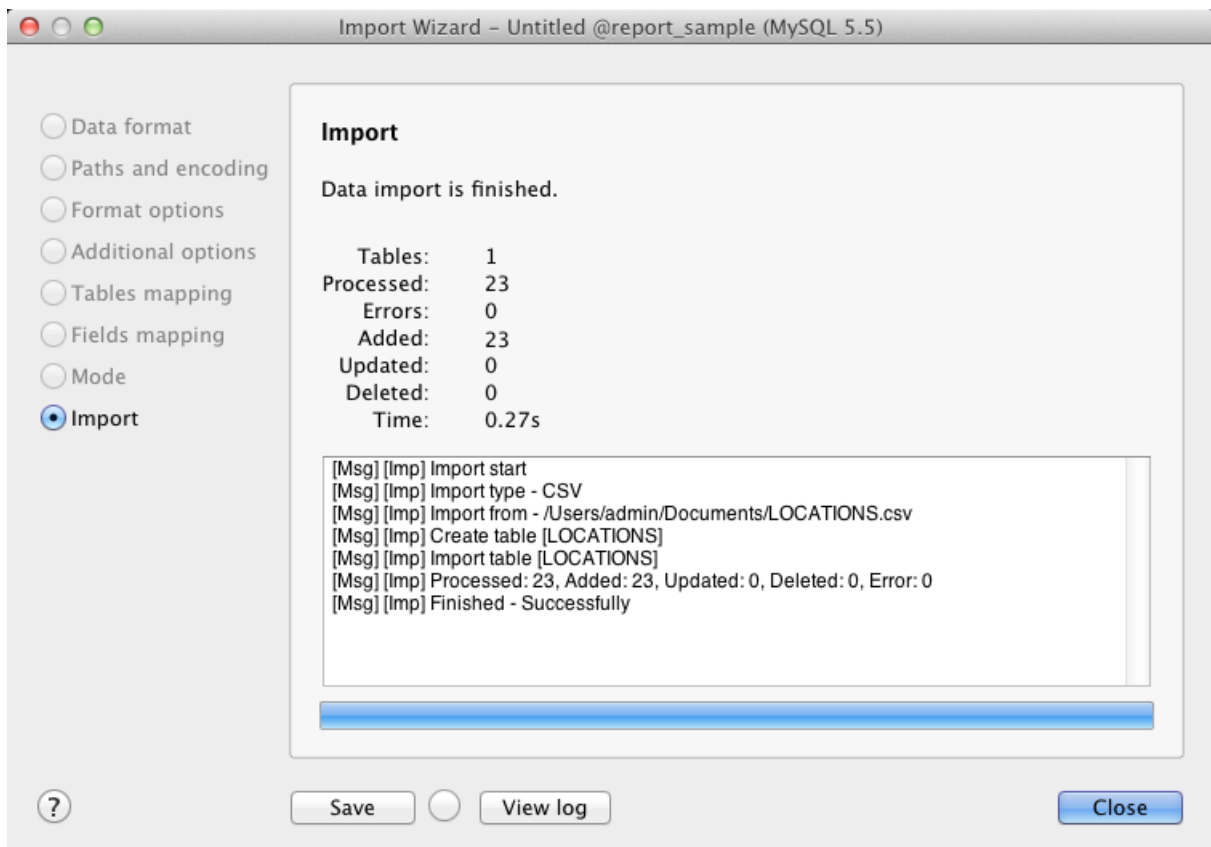
Ignores errors that are encountered during the import process.

Saving and Confirming Import

Click **Start** button to start the import process.

Hint: Click **Save** to save your settings as a profile for setting schedule.

You can click **View Log** to view the running process indicating success or failure.



Export Wizard

Export Wizard allows you to export data from table, view, or query results to any available format. You can save your settings as a profile for setting schedule.

Note: Navicat Essentials version supports to export text-based files, such as TXT, CSV and XML file.

To open the Export Wizard, select a table and click **Export** from the table object pane toolbar.

- [Setting Export File Format](#)
- [Setting Destination File Paths and Encoding](#)
- [Selecting Table Columns for Export](#)
- [Setting Additional Options](#)
- [Saving and Confirming Export](#)

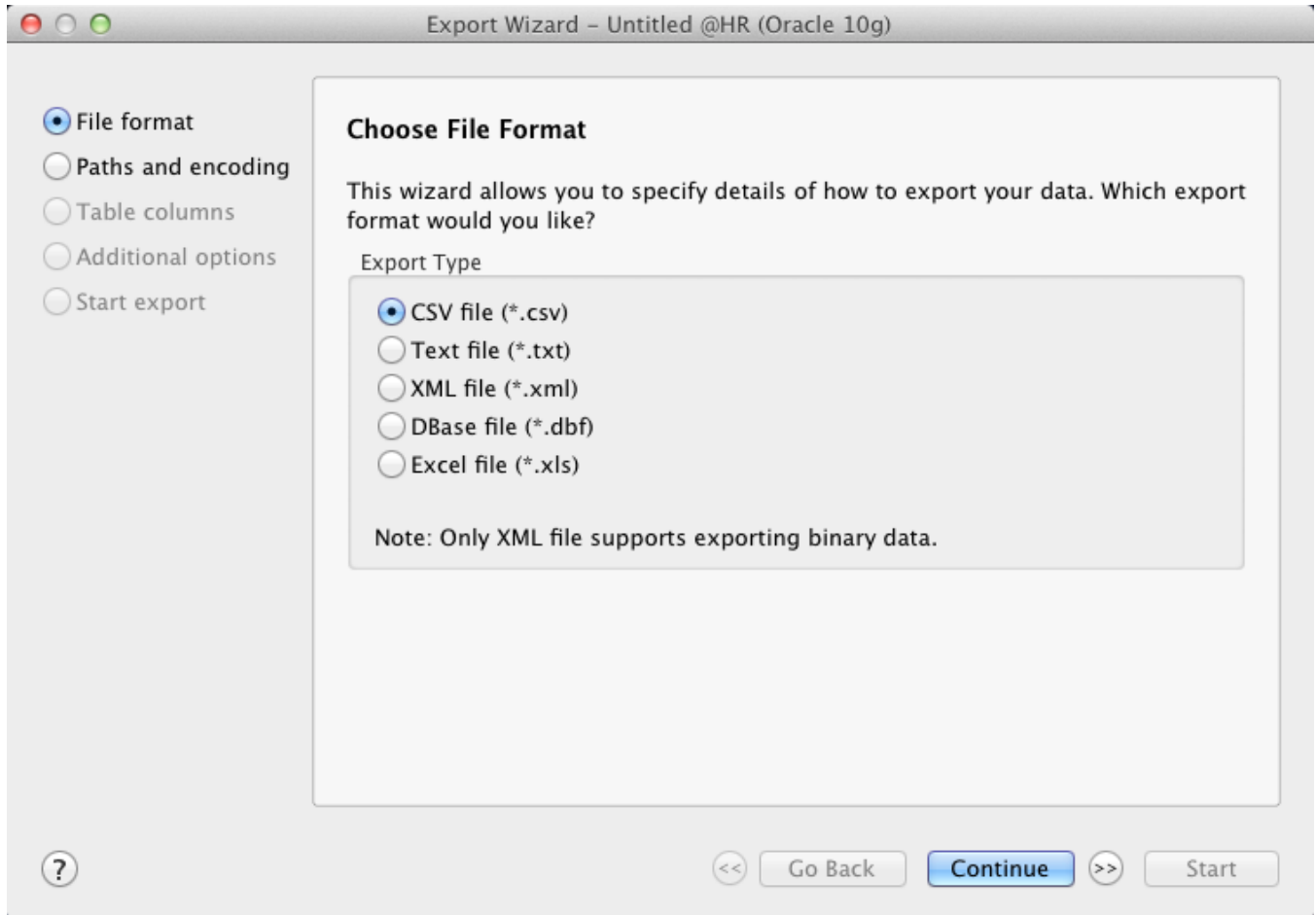
To run a saved export profile from the command line

- Create and save the export profile.
- In terminal, type the command (see Command for details)

Setting Export File Format

Select one of the available table formats.

Note: Navicat Essentials only supports exporting to TXT, CSV and XML file.



Setting Destination File Paths and Encoding

Set name for the result file. The file name extension in the **Export To** text box changes according to the selected table type in step Export File Format.

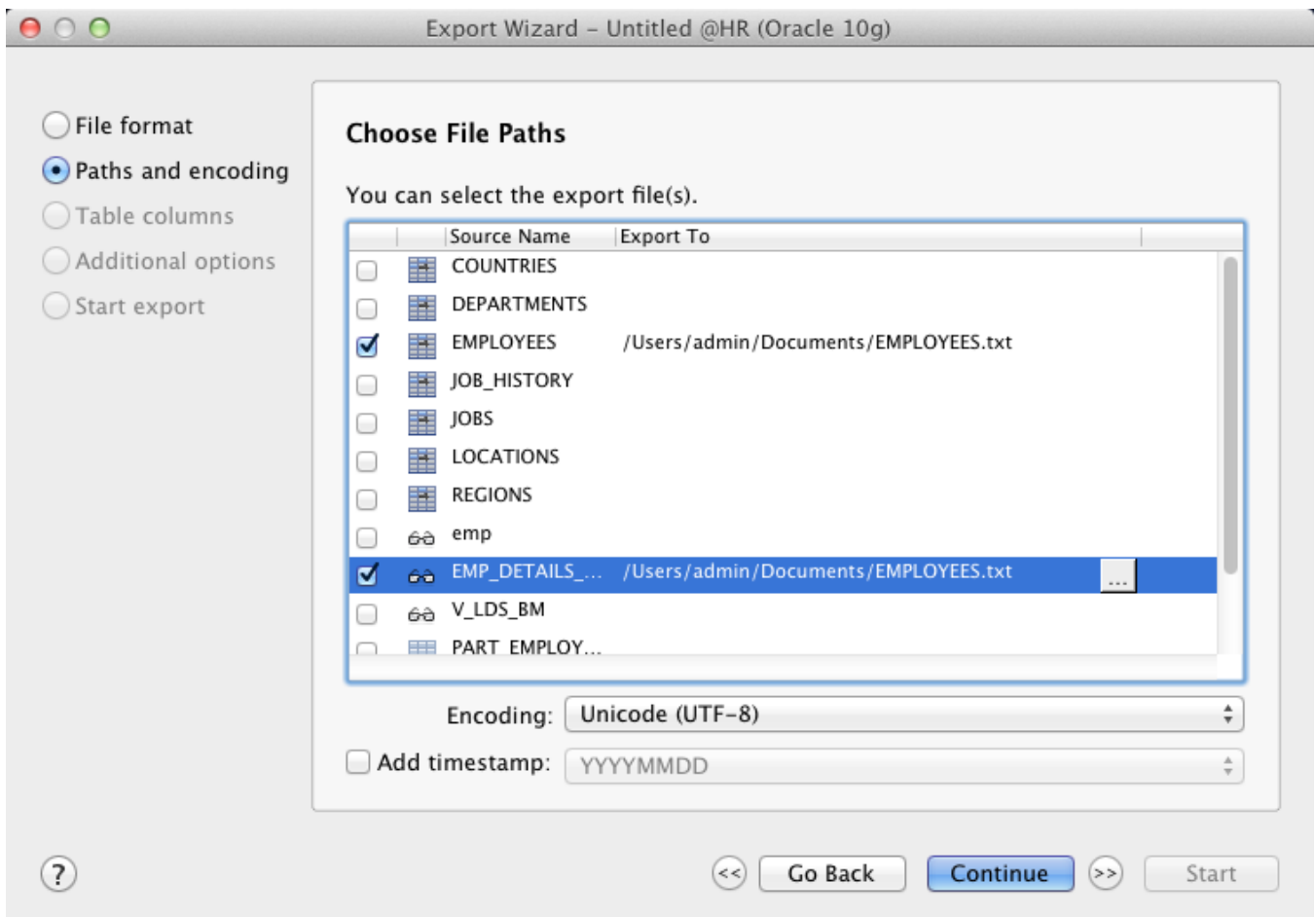
Note: For exporting query result, please ensure that you have saved the query before running the Export Wizard. Otherwise, no source table displayed in here.

Encoding

Select the encoding for the exported file.

Add Timestamp

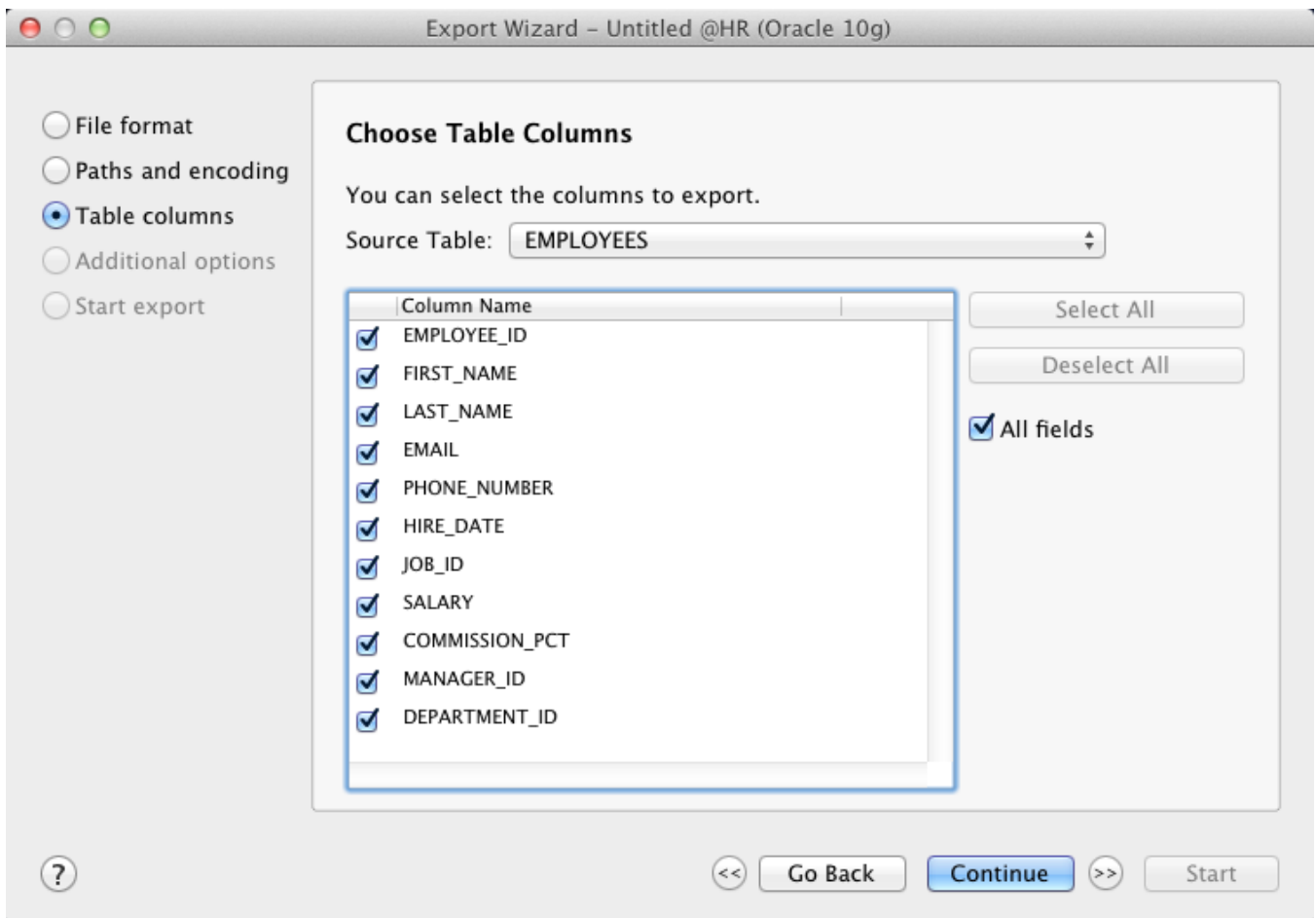
Checks this option if you want your file name specifies the timestamp of the export is run. Select the date/time format from the drop-down list.



Selecting Table Columns for Export

Select table fields for export. All the fields are selected by default. If you want to omit some fields to be exported, just simply uncheck the box **All Fields** first and then uncheck those fields in the Available Fields list.

Note: For exporting query result, the wizard will skip this step.



Setting Additional Options

You are allowed to customize formats applied to exported data.

Misc options

Include column titles

Field names will be included into the exported file if this option is on.

Blank if zero

Outputs zero if the content of field is blank.

Append on output file(s)

Appends records to the existing file. If you set exporting multiple tables to the same target file in step Setting Destination File Paths and Encoding, checks this option to append the records.

Continue on error

Ignores errors that are encountered during the export process.

Record Delimiter

Selects the record delimiter.

Text Qualifier

Selects the text qualifier.

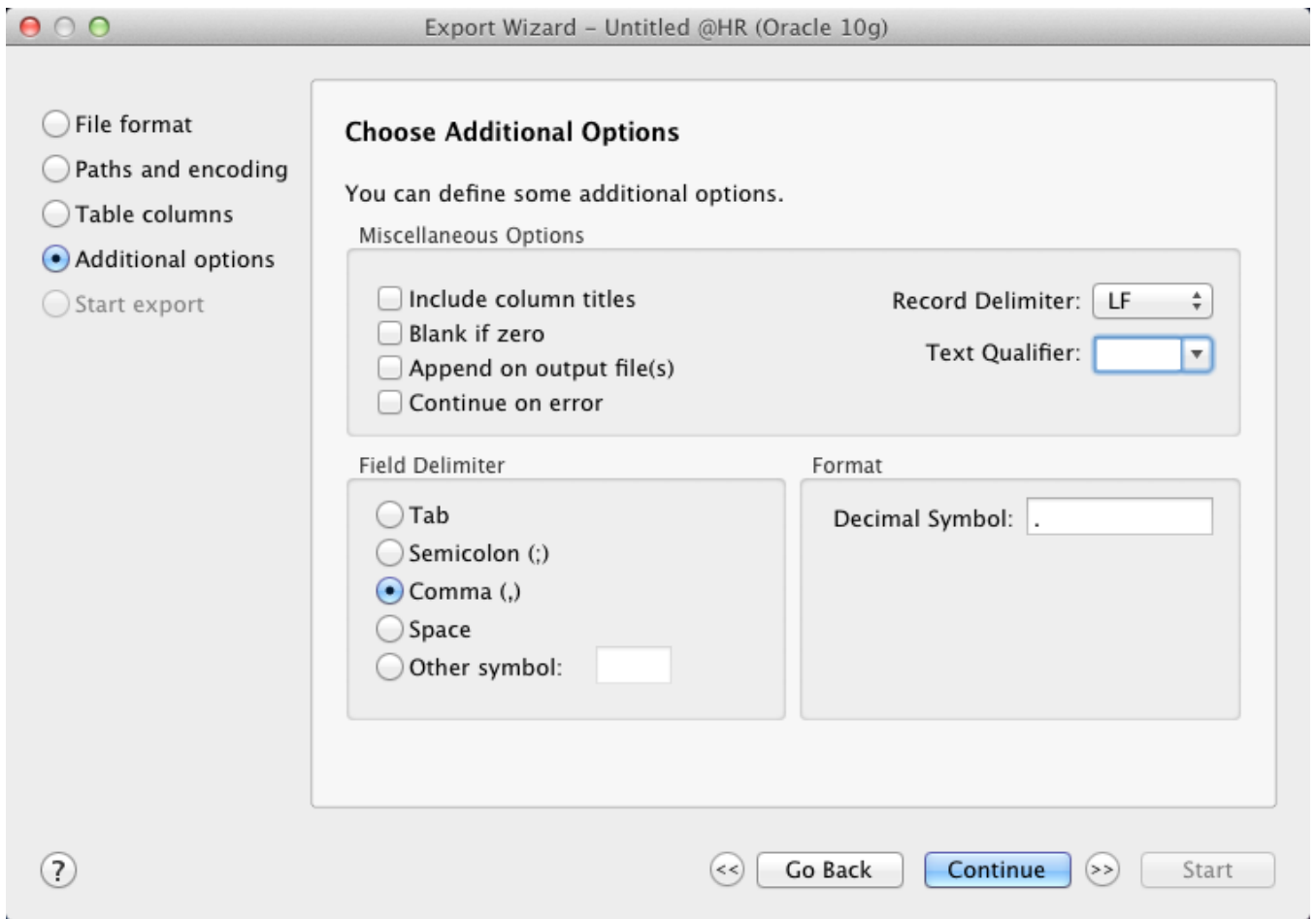
Field Delimiter

Selects the field delimiters: Tab, Semicolon(;), Comma(,), Space or Other symbol

Format

Defines the formats of the date, time and number.

Hint: Only related options will be enabled according to the selected table type in step Export File Format.

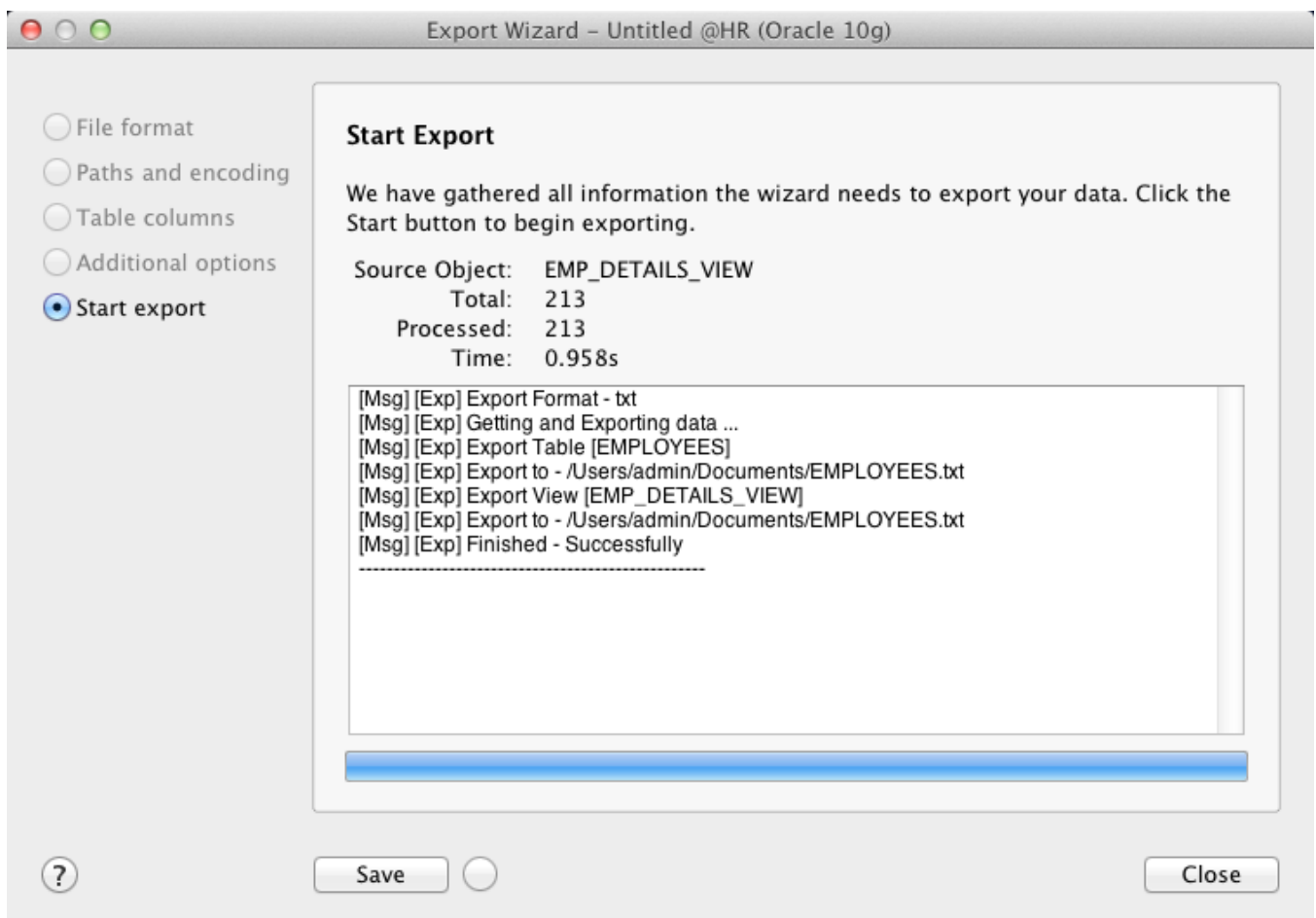


Saving and Confirming Export

Click **Start** button to start the export process.

Hint: Click **Save** to save your settings as a profile for setting schedule.

You can view the running process indicating success or failure.




Data Transfer (Available only in Full Version)

Navicat allows you to transfer tables/views/functions/sequences/events from one database/schema to another database/schema, or to a sql file. The target database/schema can be on the same server as the source database/schema or on another server. It also allows you to save a profile for easy retrieval and running of data transfer between databases/schemas. You can also invoke data transfer from the command line, which makes it possible to schedule data transfer between databases/schemas. You can save your settings as a profile for setting schedule.


Simply open the data transfer and use the data transfer toolbar, which allows you to create, save and delete the data transfer.

Create Data Transfer

To create a new data transfer

- Select **Tools** -> **Data Transfer** from the main menu or just select  **New** from the toolbar above.
- Edit data transfer properties on the appropriate tabs.

To create a new data transfer with modification as one of the existing data transfer profiles

- Select **Tools** -> **Data Transfer** from the main menu
- Select the data transfer for modifying from the **Load Profile** drop-down list.
- Modify data transfer properties on the appropriate tabs.
- Click  **Save As**.

Hint: To create new data transfer, you can also control-click the database/schema node in the navigation pane and select the **Data Transfer** from the popup menu.


Edit Data Transfer

To edit the existing data transfer

- Select **Tools** -> **Data Transfer** from the main menu.
- Select the data transfer for modifying from the **Load Profile** drop-down list.
- Modify data transfer properties on the appropriate tabs.

Run Data Transfer

To run a data transfer


- Create a new data transfer/open the existing one.
- Click  **Start**.

To run a saved data transfer profile from the command line

- Create and save data transfer profile.
- In terminal, type the command (see Command for details)

Delete Data Transfer

To delete a data transfer

- Select **Tools** -> **Data Transfer** from the main menu.
- Select the data transfer for dropping from the **Load Profile** drop-down list.
- Click the  **Delete** from the toolbar.

General Settings for Data Transfer

The following instruction guides you through the process of setting up a data transfer. Customize options according to your needs. (see drag and drop (MySQL, Oracle, PostgreSQL, SQLite or SQL Server)).

Source

Defines connection and database/schema for the source.

All the objects are selected in the **Objects** list by default. If you do not want some objects to be transferred, just simply uncheck them.

With this option is on, only the checked objects will be transferred. However, if you add any new objects in the source database/schema after you create your data transfer profile, the newly added objects will not be transferred unless you manually modify the **Objects** list.

Chooses this option if you wish all the objects being transferred to the target database/schema, all newly added objects will also be transferred without amending the data transfer profile.

Target

Connection

Transfers your selected objects directly to the other database/schema. Chooses the connection and database/schema you wish to transfer to.

File

Transfers your selected objects directly to a text file. You can select the **Server Type**, **Version** and **Encoding** for the file.

Advanced Settings for Same Server Type Data Transfer

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Include foreign key constraints

Includes foreign keys in the table with this option is on.

Include engine/table type (Available only for MySQL)

Includes table type with this option is on.

Include character set (Available only for MySQL)

Includes character set in the table with this option is on.

Include auto increment (Available only for MySQL and SQLite)

Includes auto increment in the table with this option is on.

Include unique constraints (Available only for Oracle, PostgreSQL, SQLite and SQL Server)

Includes uniques in the table with this option is on.

Include rules (Available only for PostgreSQL)

Includes rules in the table with this option is on.

Include check constraints (Available only for Oracle, PostgreSQL, SQLite and SQL Server)

Includes checks in the table with this option is on.

Include storage (Available only for SQL Server)

Includes storage related properties of the table with this option is on.

Include triggers

Includes triggers in the table with this option is on.

Include exclude constraints (Available only for PostgreSQL)

Includes exclusion constraints in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables (Available only for MySQL, PostgreSQL and SQL Server)

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Use extended insert statements (Available only for MySQL)

Inserts records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne',  
'56'), ('0', 'Katherine', '23');
```

Use delayed insert statements (Available only for MySQL)

Inserts records using *DELAYED* insert SQL statements.

Example:

```
INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindsy', '23');  
INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');  
INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');
```

Run multiple insert statements (Available only for PostgreSQL and SQL Server)

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB (Available only for MySQL, PostgreSQL, SQLite and SQL Server)

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables (Available only for MySQL, Oracle, PostgreSQL and SQL Server)

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Drop with cascade (Available only for PostgreSQL)

Check this option to drop objects with cascade.

Create target database if not exist (Available only for MySQL)

Creates a new database if the database specified in target server does not exist.

Create target database/schema if not exist (Available only for PostgreSQL and SQL Server)

Creates a new database/schema if the database/schema specified in target server does not exist.

Include schema name in statements (Available only for Oracle, PostgreSQL and SQLite)

Check this option to include the schema name in the SQL statement.

Use DDL from SHOW CREATE TABLE (Available only for MySQL)

If this option is on, DDL will be used from show create table.

Use DDL from sqlite_master (Available only for SQLite)

If this option is on, DDL will be used from the *SQLITE_MASTER* table.

Advanced Settings for Cross Server Data Transfer (Available only in Navicat Premium)

Navicat Premium supports transferring data across different server types, e.g. from MySQL to Oracle. The Data Transfer process can transfer tables with primary key constraints. The following part shows the settings for different server types.

- [Data transfer from MySQL to Oracle](#)
- [Data transfer from MySQL to PostgreSQL](#)
- [Data transfer from MySQL to SQLite](#)
- [Data transfer from MySQL to SQL Server](#)
- [Data transfer from Oracle to MySQL](#)
- [Data transfer from Oracle to PostgreSQL](#)
- [Data transfer from Oracle to SQLite](#)
- [Data transfer from Oracle to SQL Server](#)
- [Data transfer from PostgreSQL to MySQL](#)
- [Data transfer from PostgreSQL to Oracle](#)
- [Data transfer from PostgreSQL to SQLite](#)
- [Data transfer from PostgreSQL to SQL Server](#)
- [Data transfer from SQLite to MySQL](#)
- [Data transfer from SQLite to Oracle](#)
- [Data transfer from SQLite to PostgreSQL](#)
- [Data transfer from SQLite to SQL Server](#)
- [Data transfer from SQL Server to MySQL](#)
- [Data transfer from SQL Server to Oracle](#)
- [Data transfer from SQL Server to PostgreSQL](#)
- [Data transfer from SQL Server to SQLite](#)

Advanced Settings for Transferring from MySQL to Oracle

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Use single transaction (InnoDB only)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the process starts.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Advanced Settings for Transferring from MySQL to PostgreSQL

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');
```

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Use single transaction (InnoDB only)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the process starts.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Drop with cascade

Check this option to drop objects with cascade.

Create target database/schema if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from MySQL to SQLite

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Use single transaction (InnoDB only)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the process starts.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Advanced Settings for Transferring from MySQL to SQL Server

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');
```

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Use single transaction (InnoDB only)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the process starts.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Create target database/schema if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from Oracle to MySQL

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');
```

Use extended insert statements

Inserts records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

Use delayed insert statements

Inserts records using *DELAYED* insert SQL statements.

Example:

```
INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindsy', '23');  
INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');  
INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');
```

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Create target database if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from Oracle to PostgreSQL

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0',  
'katherine', '23');
```

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Drop with cascade

Check this option to drop objects with cascade.

Create target database/schema if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from Oracle to SQLite

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Advanced Settings for Transferring from Oracle to SQL Server

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Create target database/schema if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from PostgreSQL to MySQL

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0',  
'katherine', '23');
```

Use extended insert statements

Inserts records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

Use delayed insert statements

Inserts records using *DELAYED* insert SQL statements.

Example:

```
INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindsy', '23');  
INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');  
INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');
```

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Create target database if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from PostgreSQL to Oracle

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Advanced Settings for Transferring from PostgreSQL to SQLite

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Advanced Settings for Transferring from PostgreSQL to SQL Server

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0',  
'katherine', '23');
```

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Create target database/schema if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from SQLite to MySQL

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');
```

Use extended insert statements

Inserts records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

Use delayed insert statements

Inserts records using *DELAYED* insert SQL statements.

Example:

```
INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindsy', '23');  
INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');  
INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');
```

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Advanced Settings for Transferring from SQLite to Oracle

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Advanced Settings for Transferring from SQLite to PostgreSQL

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');
```

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Drop with cascade

Check this option to drop objects with cascade.

Create target database/schema if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from SQLite to SQL Server

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0',  
'katherine', '23');
```

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Create target database/schema if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from SQL Server to MySQL

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');
```

Use extended insert statements

Inserts records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindsy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

Use delayed insert statements

Inserts records using *DELAYED* insert SQL statements.

Example:

```
INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindsy', '23');  
INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');  
INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');
```

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Create target database if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from SQL Server to Oracle

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Advanced Settings for Transferring from SQL Server to PostgreSQL

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Lock target tables

Locks the tables in the target database/schema during the data transfer process.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0',  
'katherine', '23');
```

Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Drop with cascade

Check this option to drop objects with cascade.

Create target database/schema if not exist

Creates a new database/schema if the database/schema specified in target server does not exist.

Advanced Settings for Transferring from SQL Server to SQLite

Table Options

Create tables

Creates tables in the target database with this option is on.

Supposes this option is unchecked and tables already exist in the target database, then all data will be appended to the destination tables.

Include indexes

Includes indexes in the table with this option is on.

Record Options

Insert records

Check this option if you require all records to be transferred to the destination database/schema.

Use transaction

Check this option if you use transaction during the data transfer process.

Use complete insert statements

Inserts records using complete insert syntax.

Example:

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('1',  
'Peter McKindsy', '23');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('2',  
'Johnson Ryne', '56');
```

```
INSERT INTO `users` (` ID Number`, ` User Name`, ` User Age`) VALUES ('0',  
'katherine', '23');
```

Use hexadecimal format for BLOB

Inserts BLOB data as hexadecimal format.

Other Options

Continue on error

Ignores errors that are encountered during the transfer process.

Lock source tables

Locks the tables in the source database so that any update on the table is not allowed once the data transfer is triggered off.

Drop target objects before create

Check this option if objects already exist in the target database/schema, the existing objects will be deleted once the data transfer starts.

Data Transfer Message Log

The **Message Log** tab allows you to view the running process indicating success or failure.

Example:

```
[Msg] [Dtf] Datatransfer started
[Msg] [Dtf] Gather Information
[Msg] [Dtf] Drop table: items
[Msg] [Dtf] Create table: items
[Msg] [Dtf] Get table data for: items
[Msg] [Dtf] Begin transaction on target server
[Msg] [Dtf] Start transfer data for table: items
[Msg] [Dtf] End transaction on target server
[Msg] [Dtf] Drop view: view_for_mysql1
[Msg] [Dtf] Create view: view_for_mysql1
[Msg] [Dtf] Drop procedure: procedure1
[Msg] [Dtf] Create procedure: procedure1
[Msg] [Dtf] Finished - Successfully
```

Data Synchronization (Available only in Full Version)

Navicat allows you to transfer data from one database/schema to another database/schema with detailed analytical process. In other words, Navicat provides the ability for data in different database/schema to be kept up-to-date so that each repository contains the same information. The target database/schema can be on the same server as the source database/schema or on another server. You are not only authorized to rollback the transferring process, but also insert, delete and update records to the destination. You can also save your settings as a data synchronization profile for setting schedule. Same as Data Transfer, Data Synchronization can be invoked from the command line.

Note:

For Oracle server:


- BLOB, CLOB, NCLOB, LONG and LONG RAW data are skipped during the data synchronization process.
- TIMESTAMP primary key cannot synchronize (insert, update) with Database Link to 9i server.
- RAW primary key cannot synchronize (insert, update, delete) with Database Link to any server, without error.

Note: Only SQL Server 2005 or above supports data synchronization.


Just simply open the data synchronization and use the data synchronization toolbar, allowing you to create, save and delete the data synchronization.

Create Data Synchronization

To create a new data synchronization

- Select **Tools** -> **Data Synchronization** from the main menu or just select  **New** from the toolbar above.
- Edit data synchronization properties on the appropriate tabs.

To create a new data synchronization with modification as one of the existing data synchronization profiles

- Select **Tools** -> **Data Synchronization** from the main menu.
- Select the data synchronization for modifying from the **Load Profile** drop-down list.
- Modify data synchronization properties on the appropriate tabs.
- Click  **Save As**.


Edit Data Synchronization

To edit the existing data synchronization

- Select **Tools** -> **Data Synchronization** from the main menu.
- Select the data synchronization for modifying from the **Load Profile** drop-down list.
- Modify data synchronization properties on the appropriate tabs.


Preview Data Synchronization

To preview a data synchronization before execution

- Create a new data synchronization/open the existing one.
- Click  **Preview**.

Run Data Synchronization

To run a data synchronization


- Create a new data synchronization/open the existing one.
- Click  **Start**.

To run a saved data synchronization profile from the command line

- Create and save the data synchronization profile.
- In terminal, type the command (see Command for details)

Delete Data Synchronization

To delete a data synchronization

- Select **Tools** -> **Data Synchronization** from the main menu.
- Select the data synchronization for dropping from the **Load Profile** drop-down list.
- Click the  **Delete** from the toolbar.

General Settings for Data Synchronization

The following instruction guides you through the process of setting up a data synchronization. Customize options according to your needs.

Note: All tables must contain primary key(s) and all table structures must be identical between the source and target (see Structure Synchronization).

Source

Defines connection and database/schema for the source.

Target

Defines database link and database/schema for the target.

Note: For Oracle server, you need to create Public/Private Database Link to the target Oracle server database before.

Source table/Target table

Only tables which contain identical table names between the source and target are mapped in the list by default. If you do not want some tables to be synchronized, simply disable them manually from the drop-down list.

Hint: You can preview the outcome before execution.

Advanced Settings for Data Synchronization

Mode

Insert records, Delete records, Update records

Check these options to performing such actions to the target when data are synchronized.

Others

Save synchronization queries to file

Checks this option and select the path to save all queries in a text file.

Commit transaction in case of error

Rollbacks all data when error occurs.

Ensure that table structures match (recommended) (Available only for MySQL , PostgreSQL, SQLite and SQL Server)

Checks this option if you want to synchronize data only if the table structure of two tables are matched.

Disable foreign key checks (Available only for MySQL)

Checks this option if you want to disable foreign key check when synchronizing data.

Data Synchronization Result

The **Result** tab allows you to view the running process indicating success or failure.

Summary (the expected changes in the number of rows)

It will show the expected changes in the number of rows.

Log

You can check whether there is any errors.


Structure Synchronization (Available only in Full Version & only for MySQL, Oracle, PostgreSQL and SQL Server)

Navicat allows you to compare and modify the table structures with detailed analytical process. In other words, Navicat compares tables between two databases/schemas and states the differential in structure. The target database/schema can be on the same server as the source database/schema or on another server.


Open the structure synchronization and use the structure synchronization toolbar, allowing you to create, save and delete the structure synchronization.

Create Structure Synchronization

To create a new structure synchronization

- Select **Tools** -> **Structure Synchronization** from the main menu or just select  **New** from the toolbar above.
- Edit structure synchronization properties on the General Settings tab.

To create a new structure synchronization with modification as one of the existing structure synchronization profiles

- Select **Tools** -> **Structure Synchronization** from the main menu
- Select the structure synchronization for modifying from the drop-down list.
- Modify structure synchronization properties on the General Settings tab.
- Click  **Save As**.


Edit Structure Synchronization

To edit the existing structure synchronization

- Select **Tools** -> **Structure Synchronization** from the main menu.
- Select the structure synchronization for modifying from the drop-down list.
- Modify structure synchronization properties on the General Settings tab.


Run Structure Synchronization

To run a structure synchronization

- Create a new structure synchronization/open the existing one.
- Click  **Compare** to generate a set of scripts which shows the differentiation between the databases/schemas.
- Select the scripts you want to run.
- Click **Run Queries**.

Delete Structure Synchronization

To delete a structure synchronization

- Select **Tools** -> **Structure Synchronization** from the main menu.
- Select the structure synchronization for dropping from the drop-down list.
- Click the  **Delete** from the toolbar.

General Settings for MySQL Structure Synchronization

The following instruction guides you through the process of setting up a structure synchronization. Customize options according to your needs.

Source

Defines connection and database for the source.

Target

Defines connection and database for the target.

Compare Options

Compare tables

Check this option if you want to compare tables between the source and target databases. Select/unselect the four options below:

Compare primary keys

Check this option if you want to compare table primary keys.

Compare indexes

Check this option if you want to compare indexes.

Compare foreign keys

Check this option if you want to compare table foreign keys.

Compare triggers

Check this option if you want to compare triggers.

Compare character set

Check this option if you want to compare character set of the tables.

Compare auto increment value

Check this option if you want to compare table auto increment values.

Compare views

Check this option if you want to compare views.

Compare stored routines

Check this option if you want to compare stored routines.

Compare events

Check this option if you want to compare events.

Execution Options

SQL for objects to be created

Check this option if you want to create new objects in the target database.

SQL for objects to be changed

Check this option if you want to modify object structures in the target database.

SQL for objects to be dropped

Check this option if you want to remove objects, which do not exist in the source database, from the target database.

Compare after execution

Compares tables after the synchronization is executed.

Continue on error

Ignores errors that are encountered during the synchronization process.

General Settings for Oracle Structure Synchronization

The following instruction guides you through the process of setting up a structure synchronization. Customize options according to your needs.

Source

Defines connection and schema for the source.

Target

Defines connection and schema for the target.

Compare Options

Compare tables

Check this option if you want to compare tables between the source and target schemas. Select/unselect the four options below:

Compare primary keys

Check this option if you want to compare table primary keys.

Compare foreign keys

Check this option if you want to compare table foreign keys.

Compare uniques

Check this option if you want to compare uniques.

Compare checks

Check this option if you want to compare checks.

Compare views

Check this option if you want to compare views.

Compare procedures and functions

Check this option if you want to compare procedures and functions.

Compare indexes

Check this option if you want to compare indexes.

Compare sequences

Check this option if you want to compare sequences.

Compare triggers

Check this option if you want to compare triggers.

Compare tablespace and physical attributes

Check this option if you want to compare tablespace and physical attributes.

Execution Options

SQL for objects to be created

Check this option if you want to create new objects in the target schema.

SQL for objects to be changed

Check this option if you want to modify object structures in the target schema.

SQL for objects to be dropped

Check this option if you want to remove objects, which do not exist in the source schema, from the target schema.

Drop with cascade

Check this option if you want to drop objects with cascade.

Compare after execution

Compares tables after the synchronization is executed.

Continue on error

Ignores errors that are encountered during the synchronization process.

General Settings for PostgreSQL Structure Synchronization

The following instruction guides you through the process of setting up a structure synchronization. Customize options according to your needs.

Source

Defines connection, database and schema for the source.

Target

Defines connection, database and schema for the target.

Compare Options

Compare tables

Check this option if you want to compare tables between the source and target schemas. Select/unselect the eight options below:

Compare primary keys

Check this option if you want to compare table primary keys.

Compare indexes

Check this option if you want to compare indexes.

Compare foreign keys

Check this option if you want to compare table foreign keys.

Compare uniques

Check this option if you want to compare uniques.

Compare checks

Check this option if you want to compare checks.

Compare rules

Check this option if you want to compare rules.

Compare excludes

Check this option if you want to compare exclude constraints.

Compare triggers

Check this option if you want to compare triggers.

Compare views

Check this option if you want to compare views.

Compare functions

Check this option if you want to compare functions.

Compare sequences

Check this option if you want to compare sequences.

Execution Options

SQL for objects to be created

Check this option if you want to create new objects in the target schema.

SQL for objects to be changed

Check this option if you want to modify object structures in the target schema.

SQL for objects to be dropped

Check this option if you want to remove objects, which do not exist in the source schema, from the target schema.

Drop with cascade

Check this option if you want to drop objects with cascade.

Compare after execution

Compares tables after the synchronization is executed.

Continue on error

Ignores errors that are encountered during the synchronization process.

General Settings for SQL Server Structure Synchronization

The following instruction guides you through the process of setting up a structure synchronization. Customize options according to your needs.

Source

Defines connection, database and schema for the source.

Target

Defines connection, database and schema for the target.

Compare Options

Compare Tables

Check this option if you want to compare tables between the source and target databases. Select/unselect the four options below:

Compare primary keys

Check this option if you want to compare table primary keys.

Compare foreign keys

Check this option if you want to compare table foreign keys.

Compare uniques

Check this option if you want to compare uniques.

Compare checks

Check this option if you want to compare checks.

Compare collation

Check this option if you want to compare collation of the tables.

Compare identity last value

Check this option if you want to compare table identity last values.

Compare views

Check this option if you want to compare views.

Compare stored routines

Check this option if you want to compare stored routines.

Compare indexes

Check this option if you want to compare indexes.

Compare triggers

Check this option if you want to compare triggers.

Compare storages

Check this option if you want to compare storages.

Execution Options

SQL for objects to be created

Check this option if you want to create new objects in the target database.

SQL for objects to be changed

Check this option if you want to modify object structures in the target database.

SQL for objects to be dropped

Check this option if you want to remove objects, which do not exist in the source database, from the target database.

Compare after execution

Compares tables after the synchronization is executed.

Continue on error

Ignores errors that are encountered during the synchronization process.

Structure Synchronization Result

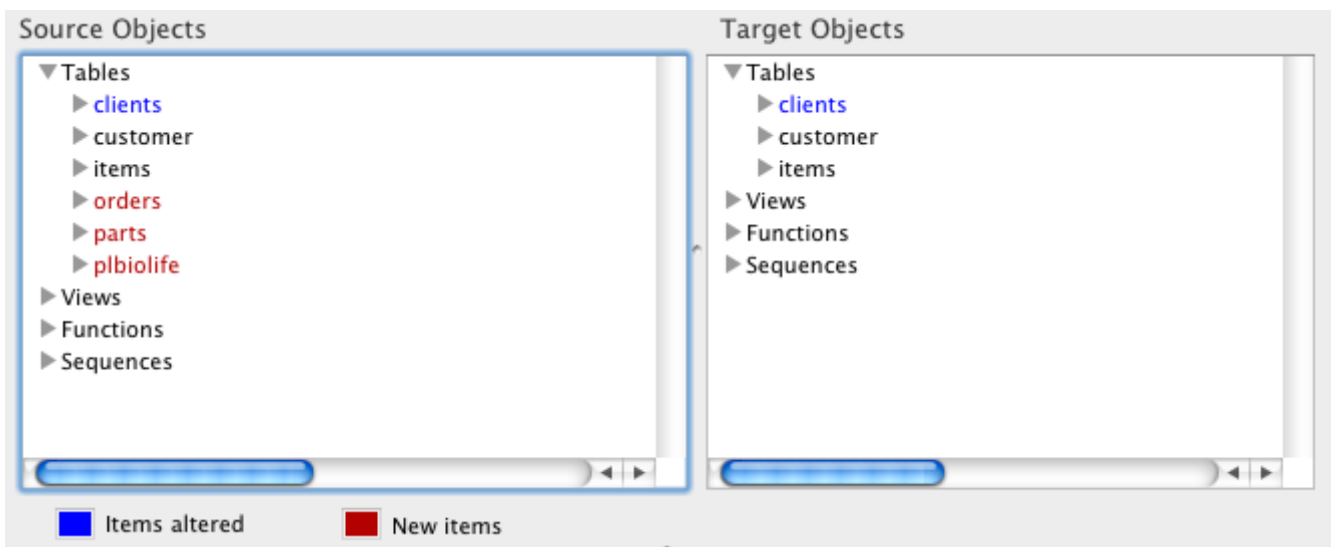
Source Objects/Target Objects

The tree view shows the differentiation between the source database/schema and target database/schema after the computation of the structure synchronization, providing with the detailed SQL statements shown in the **Result** list.

The red item represents the non-existence for the other database/schema.

The blue item represents the existence for the other database/schema, but different definition detected.

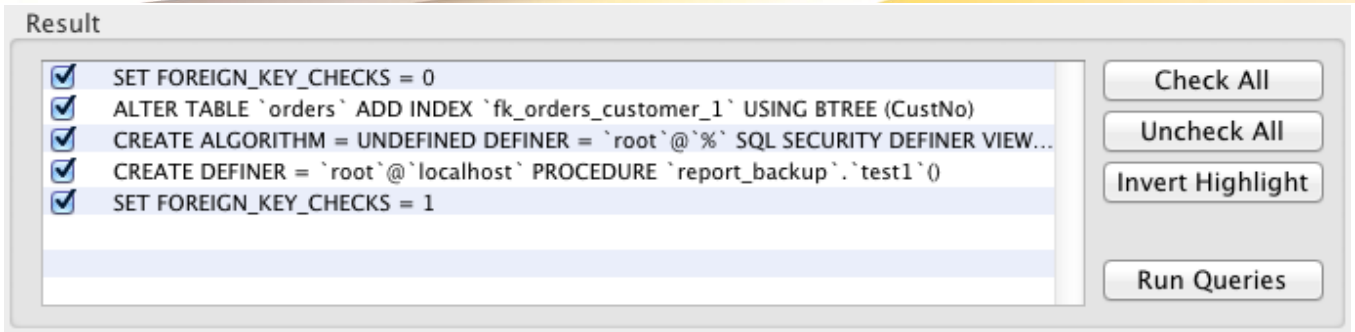
You are allowed to edit the object structure manually, simply click the object in the tree view to edit.



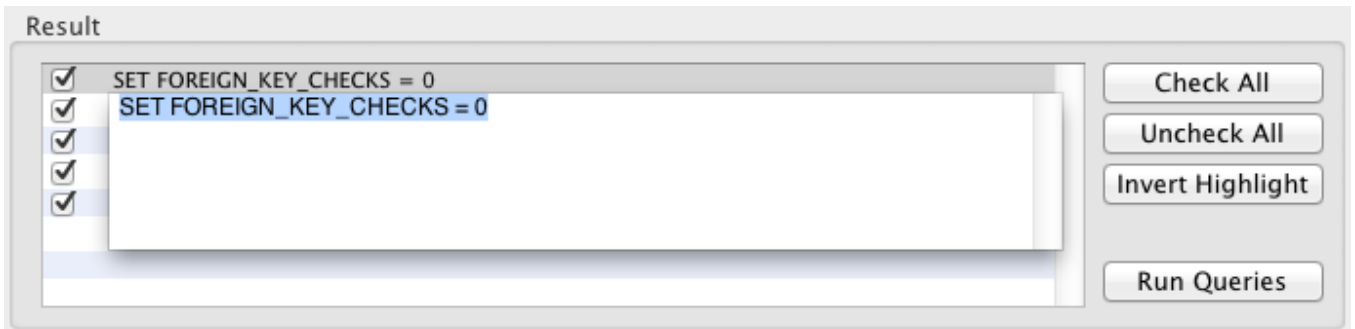
Result

All the scripts are applied to the target database/schema and they are being checked in the **Result** list by default. To execute the checked scripts, press **Run Queries**.

You can highlight multiple lines of scripts, and choose **Invert Highlighted** so as to toggle the selection status of scripts at one go.



To view the full SQL statements, simply click on the scripts.



Structure Synchronization Message Log

The **Message Log** tab allows you to view the running process indicating success or failure.

Example:


(Run Queries)

```
Query OK: ALTER TABLE 'clients' CHANGE COLUMN 'RecordID' 'RecordID' int(10)
DEFAULT NULL first
```

Backup/Restore (Available only in Full Version & only for MySQL, PostgreSQL and SQLite)


A secure and reliable server is closely related to performing regular backups as failures will probably occur sometimes - caused by attacks, hardware failure, human error, power outages, etc.

Navicat allows you to backup/restore all tables and records, views, stored procedures and events for your MySQL, PostgreSQL and SQLite databases. Backup can be invoked from the command line, which makes it possible to schedule backups between databases/schemas.

Just simply click  to open an object pane for **Backup**. A control-click displays the popup menu or using the object pane toolbar, allowing you to create new, restore, extract and delete the backup.

Create Backup

To create a new backup

- Select anywhere on the object pane.
- Click the  from the object pane toolbar.
or
- Control-click and select **New Backup** from the popup menu.
- Edit backup properties on the appropriate tabs.
- Click **Start**.

Hint: To create new backup you can also control-click the Backups node of the navigation pane and select the **New Backup** from the popup menu.


Edit Backup

To change the name of the backup

- Select the backup for editing in the navigation pane/object pane.
- Control-click and select the **Rename** from the popup menu.

Delete Backup

To delete a backup

- Select the backup for deleting in the navigation pane/object pane.
- Control-click and select the **Delete Backup** from the popup menu.
or
- Click the  from the object pane toolbar.
- Confirm deleting in the dialog window.


Restore Backup

To restore a backup to an existing database

- Open the database and select the existing backup.
- Click **Open** from the object pane toolbar or control-click and select **Restore Backup** from the popup menu.
- Click **Start**.

Hint: To restore the backup you can also click **Open** from the object pane toolbar or control-click the Backup node of the navigation pane and select the **Restore Backup** from the popup menu.

To restore a backup to a new database

- Create a new database (MySQL or PostgreSQL or SQLite) and click  to open an object pane for Backup.
- Click **Open** from the object pane toolbar or control-click anywhere on the object pane and select **Restore Backup** from the popup menu.
- Browse the backup file.
- Click **Start**.


Extract SQL

To extract backup to sql file

- Select the backup for extracting in the navigation pane/object pane.
- Control-click and select the **Extract SQL** from the popup menu.
- Save the sql file to your destination.

Achieve Backup Information

To achieve a backup information (Name, File Size and Update Time etc)

- Select the backup in the navigation pane/object pane.
- Choose View -> Object Information in the main menu.
or
- Click the  from the object pane toolbar.

Backup

Backup is the basic Navicat tool for performing regular backups.

- [General Settings for Backup](#)
- [Object Selection for Backup](#)
- [Advanced Settings for Backup](#)
- [Message Log](#)

Hint: Backup files are stored under Settings Save Path.

To run a backup from the command line

- Create and save the backup profile.
- In terminal, type the command (see Command for details)

General Settings for Backup

Allows you to view the information for the backup, including Connection Name, Host, Login User, Database and Schema.

Object Selection for Backup

You are allowed to choose your preferable database objects, i.e. tables, views, functions, sequences, events, indexes and triggers you wish to backup.

Advanced Settings for Backup

Lock all tables

Locks all objects while backup is being processed.

Use single transaction (InnoDB only) (Available only for MySQL)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the backup process starts.

Use specified file name

Defines your file name for backup. Otherwise, your backup file will be named as "**20070510173820**" for example.

Backup Message Log

The **Message Log** tab allows you to view the running process indicating success or failure.

Example:

```
[Msg] [Bak] Starting backup...
[Msg] [Bak] Gather Information
[Msg] [Bak] Get table data for clients
[Msg] [Bak] Drop table: clients
[Msg] [Bak] Create table: clients
[Msg] [Bak] Begin transaction on target server
[Msg] [Bak] Start transfer data for table: clients
[Msg] [Bak] End transaction on target server
[Msg] [Bak] Finished - Successfully
```

Restore

Navicat provides you a useful tool for restoring your backup while hardware failure occurs.

- [General Settings for Restore](#)
- [Message Log](#)

Note: You must have Create, Drop and Insert Privileges (MySQL or PostgreSQL) to run the restore.

Hint: Restore function will firstly drop the selected objects of the database, then recreate the new objects according to your backup. Finally, inserting the data.

To restore a backup with default settings from the command line

- In terminal, type the command (see Command for details)

General Settings for Restore

General tab shows the backup information, including Connection Name, Host, Login user, Database and Schema.

Restore Message Log

The **Message Log** tab allows you to view the running process indicating success or failure.

Example:


```
[Msg] Starting restore...  
[Msg] Reading Backup...  
[Msg] Restore Backup...  
[Msg] Finished successfully
```

Extract SQL

You are allowed to extract your backup into a SQL file.


Batch Job/Schedule (Available only in Full Version)

Navicat allows you to create a batch job for setting schedule to execute at one regular interval. Batch job can be created for Query, Data Transfer, Data Synchronization, Import and Export from MySQL, Oracle, PostgreSQL, SQLite and SQL Server. You can define a list of actions to be performed within one batch job, either run it manually or at the specified time/periodically.



Just simply click  to open an object pane for **Schedule**. A control-click displays the popup menu or using the schedule object pane toolbar, allowing you to create new, edit, open and delete the selected batch job/schedule.

Create Batch Job

To create a new batch job

- Select anywhere on the object pane.
- Click the  from the object pane toolbar.
or
- Control-click and select **New Batch Job** from the popup menu.
- Edit batch job properties.

To create a new batch job with modification as one of the existing batch jobs


- Select the batch job for modifying in the object pane.
- Control-click and select the **Design Batch Job** from the popup menu.
or
- Click the  from the object pane toolbar.
- Modify batch job properties.
- Click  **Save As**.

To duplicate an existing batch job

- Select the batch job for modifying in the object pane.
- Control-click and select the **Duplicate Batch Job** from the popup menu.
- The newly created batch job will be named as "batchjobname_**copy**".

Edit Batch Job

To edit the existing batch job


- Select the batch job for editing in the object pane.
- Control-click and select the **Design Batch Job** from the popup menu.
or
- Click the  **Design Batch Job** from the object pane toolbar.
- Edit batch job properties.

To change the name of the batch job

- Select the batch job for editing in the object pane.
- Control-click and select the **Rename** from the popup menu.

Run Batch Job

To run a batch job


- Create a new batch job/open the existing one.
- Click  **Run**.

To run from Command Line

- Create and save profile.
- In terminal, type the command (see Command for details)

Delete Batch Job

To delete a batch job

- Select the batch job for deleting in the object pane.
- Control-click and select the **Delete Batch Job** from the popup menu.
or
- Click the  from the object pane toolbar.
- Confirm deleting in the dialog window.

Convert Batch Job

To convert a batch job

- Control-click and select the **Batch Job Converter** from the popup menu in the object pane.
- Select the batch jobs.
- Set the convert options.
- Click **Start**.

Set Schedule

To set schedule to the batch job

- Create and save the batch job/open the existing one.
- Click the **Schedule** from the object pane toolbar or control-click and select the **Setup Run Schedule** from the popup menu.
- Set your schedule of the batch job.


Hint: CmdLineJob.log stores all the operations executed, indicating success or failure during the schedule.

To show all set schedules

- In terminal, type the command (see Command for details)

Delete Schedule

To delete a schedule


- Select the scheduled batch job in the object pane.
- Control-click and select the **Erase Schedule** from the popup menu.
or
- Click the  **Erase Schedule** from the toolbar.

General Settings for Batch Job/Schedule

The following instruction guides you through the process of setting up a batch job/schedule. Customize options according to your needs.

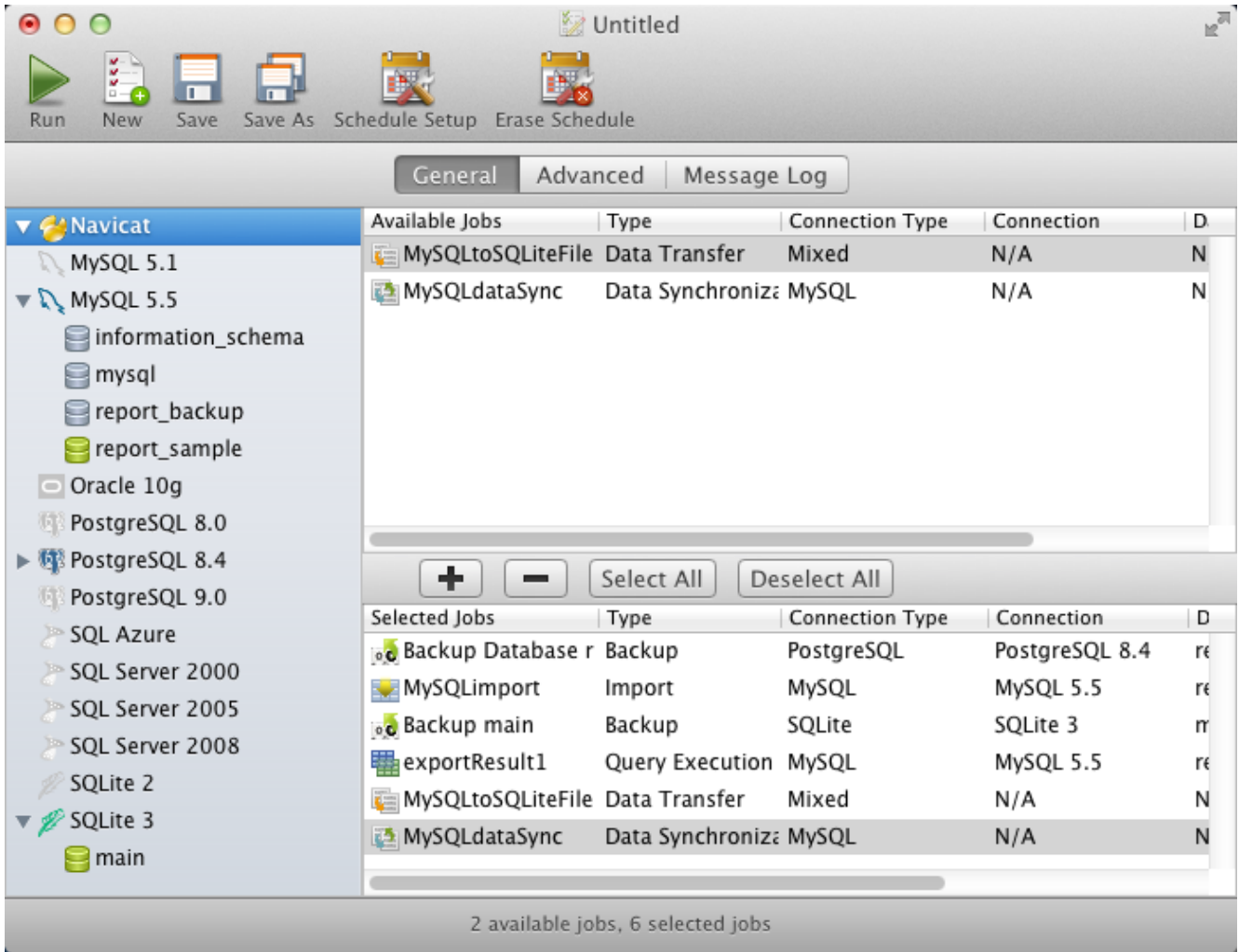
Move the objects from the **Available Jobs** list to the **Selected Jobs** list using **+**, **Select All** button or by double-click them. To delete the objects from the selected jobs list, remove them using **-**, **Deselect All** button. You are allowed to run profiles from different servers in a single batch job/schedule.

You are allowed to group jobs into a procedure, running them in sequence, each starting the next. To rearrange the sequence of the jobs, simply drag it to the desired location.

To set schedule for running Data Transfer or Data Synchronization profile, choose  **Navicat** at the top on the left panel.

Note: Please save the batch job before setting schedule.

<p>Note: Passwords must be saved in Connection Properties (MySQL, Oracle, PostgreSQL or SQL Server) before running your schedule.</p>	<h3>Connection Properties</h3> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px;"> <p>User Name: <input type="text" value="root"/></p> <p>Password: <input type="password" value="....."/> <input checked="" type="checkbox"/> Save password</p> </div>
--	--



Advanced Settings for Batch Job/Schedule

Navicat allows you to generate and send personalized mails with results returned from a schedule. The resultset(s) can be emailed to multiple recipients.

Send Email

From name

Specifies the sender's name.

From address

Specifies the e-mail address that people should use when sending e-mail to you at this account. For example, someone@navicat.com.

To, CC

Specifies the e-mail address of each recipient, separating them with a comma or a semicolon (;).

Send Attachment

Checks this option to attach file output from Export Wizard or Data Transfer.

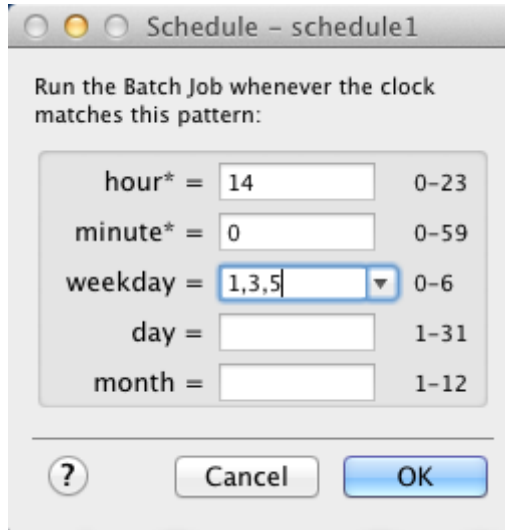
Send test mail

Navicat will send you a test mail indicating success or failure.

Setup Schedule

Schedule Setting Examples

Example 1 : The batch job will be executed at 2:00pm on every Monday, Wednesday and Friday.

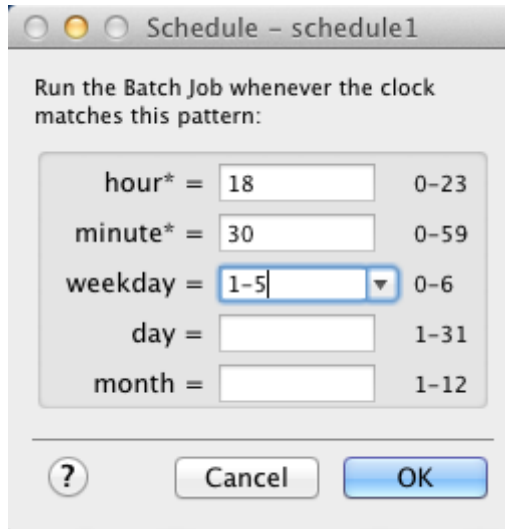


Screenshot of the "Schedule - schedule1" dialog box. The dialog contains the following fields and values:

Field	Value	Range
hour*	14	0-23
minute*	0	0-59
weekday	1,3,5	0-6
day		1-31
month		1-12

Buttons: ? (Help), Cancel, OK

Example 2 : The batch job will be executed at 6:30pm every weekday.

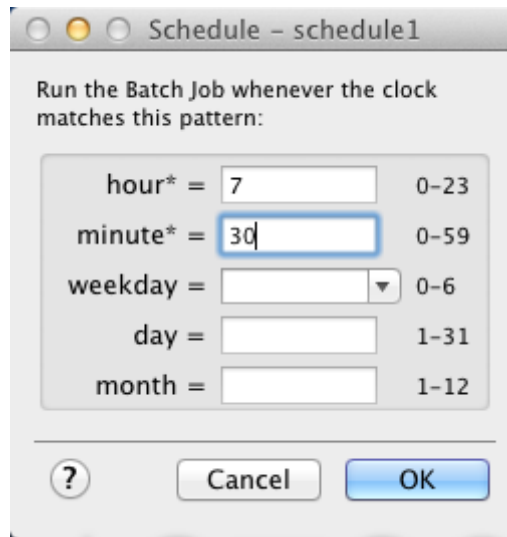


Screenshot of the "Schedule - schedule1" dialog box. The dialog contains the following fields and values:

Field	Value	Range
hour*	18	0-23
minute*	30	0-59
weekday	1-5	0-6
day		1-31
month		1-12

Buttons: ? (Help), Cancel, OK

Example 3 : The batch job will be executed at 7:30am everyday.



Note:

- If a field is left without a value, then all the values will be used. For example, if the "weekday" field is empty, then the system will treat the field to be entered with "0, 1, 2, 3, 4, 5, 6".
- Use commas to separate values. For example, "0, 1, 3, 6"
- Use hyphen, without spaces to indicate values. For example, "0-4".

Remarks: The pattern follows a crontab entry's format.

Tips: What is crontab?

For commands that need to be executed repeatedly (e.g., hourly, daily, or weekly), you can use the crontab command. The crontab command creates a crontab file containing commands and instructions for the cron daemon to execute. You can use the crontab command with the following options:

crontab -a filename	Install filename as your crontab file. On many systems, this command is executed simply as crontab filename (i.e., without the -a option).
crontab -e	Edit your crontab file, or create one if it does not already exist.
crontab -l	Display your crontab file.

crontab -r	Remove your crontab file.
crontab -v	Display the last time you edited your crontab file. (This option is only available on a few systems.)
crontab -u user	Used in conjunction with other options, this option allows you to modify or view the crontab file of user. When available, only administrators can use this option.

Each entry in a crontab file consists of six fields, specifying in the following order:

minute(s) hour(s) day(s) month(s) weekday(s) command(s)

The fields are separated by spaces or tabs. The first five are integer patterns and the sixth is the command to execute. The following table briefly describes each of the fields:

Field	Value	Description
minute	0-59	The exact minute that the command sequence executes
hour	0-23	The hour of the day that the command sequence executes
day	1-31	The day of the month that the command sequence executes
month	1-12	The month of the year that the command sequence executes
weekday	0-6	The day of the week that the command sequence executes (Sunday = 0, Monday = 1, Tuesday = 2, and so forth)
command	Special	The complete sequence of commands to execute. The command string must conform to Bourne shell syntax. Commands, executables (such as scripts), or combinations are acceptable.

Batch Job/Schedule Message Log

The **Message Log** tab allows you to view the running process indicating success or failure.

Example:

```
[2010-03-16 16:30:54 +0800] ===== Navicat Batch Job Start =====
[2010-03-16 16:30:55 +0800] Backup started (with connection-name= `Localhost`,
database= `report_sample`, schema= `report_sample`)
[2010-03-16 16:30:55 +0800] Gather Information
[2010-03-16 16:30:55 +0800] Backup finished successfully
[2010-03-16 16:30:54 +0800] ===== Navicat Batch Job Start =====
[2010-03-16 16:30:55 +0800] Exported Started (with connection-name=
[2010-03-16 16:30:55 +0800] Gather Information
[2010-03-16 16:30:55 +0800] Backup finished successfully `Localhost`,
database= `report_sample`, profile-name= `export_1`)
[Msg] [Exp] Export Format - xls
[Msg] [Exp] Getting and Exporting data...
[Msg] [Exp] Export Table [items]
[Msg] [Exp] Export to - /Users/siuha/Desktop/items.xls
[Msg] [Exp] Data Ready, Dumping to File...
[Msg] [Exp] Finished - Successfully
[2010-03-16 16:30:55 +0800] Export ended
```

Batch Job Converter (Available only for Navicat Premium)

Navicat Premium allows you to convert saved batch jobs from Navicat for MySQL, Navicat for Oracle, Navicat for PostgreSQL, Navicat for SQLite and Navicat for SQL Server to it.

A control-click in **Schedule** object pane and select **Batch Job Converter** from the popup menu to open the batch job converter window.

- [Selecting batch jobs](#)
- [Setting convert options](#)
- [Starting convert](#)

Selecting Batch Jobs

Select batch jobs to convert.

Select All

You can select all batch jobs by simply selecting **Select All** button for quick mapping.

Unselect All

You can unselect all selected batch jobs easily.

Setting Convert Options

Options

Create schedules

Check this option to copy the existing schedule of selected batch jobs in Navicat to Navicat Premium.

Delete jobs from original application

Check this option if you want to delete the original batch jobs in Navicat. If the original batch job is deleted, the scheduled batch job will not work until it is set again in Navicat Premium or the original application.

Overwrite existing jobs

Check this option if you want to overwrite the existing batch jobs in Navicat Premium.

Starting Convert

You can view the running process indicating success or failure.

Example:

```
[2010-03-19 13:07:54 +0800] ===== Batch Job Convert Start =====  
[2010-03-19 13:07:54 +0800] Created: 8.1_mysql_schedule1  
Created schedules: 1  
Overwritten schedules: 0  
Deleted schedules: 0  
[2010-03-19 13:07:54 +0800] ===== Batch Job Convert Ended =====
```

Console

Navicat provides interactive text-based screen for user query input and result output from MySQL, Oracle, PostgreSQL, SQLite and SQL Server.

- [MySQL Console](#)
- [Oracle Console](#)
- [PostgreSQL Console](#)
- [SQLite Console](#)
- [SQL Server Console](#)

MySQL Console

MySQL Console allows you to use a command-line interface. In other words, it provides interactive text-based screen for you to query input and result output from database.

Open Console

To open Console

- Open the connection and select **Tools** -> **Console** from the main menu or just simply press **Cmd-Shift-C**.
- Edit your SQL statement in the console command prompt.

Hint: To open a new console window you can also control-click the Connection/Database node of the navigation pane and select **Console** from the popup menu.

Hint: You are allowed to open multiple console windows which represents different connection each.

Exit Console

To exit Console

- Click the cross button at the main bar.

Example of Using MySQL Console

Basic MySQL query statements

```
mysql> show databases;
```

```
+-----+  
| Database          |  
+-----+  
| information_schema |  
| mysql             |  
| report_backup     |  
| report_sample     |  
+-----+
```

```
4 rows in set (0.01 sec)
```

```
mysql> use report_sample;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> show tables;
```

```
+-----+  
| Tables_in_report_sample |  
+-----+  
| clients                 |  
| customer                |  
| items                   |  
| orders                  |  
| parts                   |  
| plbiolife               |  
+-----+
```

```
6 rows in set (0.01 sec)
```

```
mysql> select Description, Cost from parts where PartNo = 2619;
```

```
+-----+-----+  
| Description          | Cost |  
+-----+-----+  
| Navigation Compass | 9.177 |  
+-----+-----+
```

```
1 row in set (0.03 sec)
```

Oracle Console

Oracle Console (SQL*Plus) allows you to use a command-line interface. In other words, it provides interactive text-based screen for you to query input and result output from database.

Note: You have to have SQL*Plus executable in order to get this works. If it is not set under the SQL*Plus default path, you are prompted to locate the executable in the pop up.

Downloading

SQL*Plus is included in packages **Oracle Client** / **Oracle Instant Client**. You can download it through -

Oracle Client

<http://www.oracle.com/technology/software/products/database/index.html>

Oracle Instant Client

<http://www.oracle.com/technology/software/tech/oci/instantclient/index.html>

Open Console

To open console

- Open the connection and select **Tools** -> **Console** from the main menu or just simply press **Cmd-Shift-C**.
- Edit your SQL statement in the console command prompt.

Hint: To open a new console window you can also control-click the Connection/Schema node of the navigation pane and select **SQL*Plus** from the popup menu.

Hint: You are allowed to open multiple console windows which represents different connection each.

Exit Console

To exit console

- Click the cross button at the main bar.
or
- Type **exit** or **quit** in the console window and press Enter.

Example of using SQL*Plus

Basic Oracle SQL query statements

```
SQL> select DEPARTMENT_ID, DEPARTMENT_NAME from DEPARTMENTS where  
LOCATION_ID = 1700;
```

```
DEPARTMENT_ID DEPARTMENT_NAME
```

```
-----
```

```
10 Administration  
90 Executive  
100 Finance  
110 Accounting  
120 Treasury  
270 Payroll
```

```
6 rows selected.
```

```
SQL>
```

PostgreSQL Console

PostgreSQL Console allows you to use a command-line interface. In other words, it provides interactive text-based screen for you to query input and result output from database.

Open Console

To open Console

- Open the connection and select **Tools** -> **Console** from the main menu or just simply press **Cmd-Shift-C**.
- Edit your SQL statement in the console command prompt.

Hint: To open a new console window you can also control-click the Connection/Database/Schema node of the navigation pane and select **Console** from the popup menu.

Hint: You are allowed to open multiple console windows which represents different connection each.

Exit Console

To exit Console

- Click the cross button at the main bar.

Example of Using PostgreSQL Console

Basic PostgreSQL query statements

```
report_sample=# select datname from pg_database;
```

```
+-----+
| datname      |
+-----+
| template1    |
| template0    |
| postgres     |
| report_sample|
| report_backup|
+-----+
```

5 rows in set (0.01 sec)

```
report_sample=# select tablename from pg_tables;
```

```
+-----+
| tablename          |
+-----+
| customer           |
| clients            |
| items              |
| orders             |
| parts              |
| plbiolife          |
+-----+
```

6 rows in set (0.02 sec)

```
report_sample=# select "Description", "Cost" from public.parts where "PartNo" = 2619;
```

```
+-----+-----+
| Description      | Cost |
+-----+-----+
| Navigation Compass | 9.177 |
+-----+-----+
```

1 row in set (0.03 sec)

```
report_sample=#
```

SQLite Console

SQLite Console allows you to use a command-line interface. In other words, it provides interactive text-based screen for you to query input and result output from database.

Open Console

To open Console

- Open the connection and select **Tools** -> **Console** from the main menu or just simply press **Cmd-Shift-C**.
- Edit your SQL statement in the console command prompt.

Hint: To open a new console window you can also control-click the Connection/Database node of the navigation pane and select **Console** from the popup menu.

Hint: You are allowed to open multiple console windows which represents different connection each.

Exit Console

To exit Console

- Click the cross button at the main bar.

Example of Using SQLite Console

Basic SQLite query statements

```
sqlite> pragma database_list;
+---+-----+-----+
| 0 | main | /Users/siuha/Desktop/sqlite_table.sqlite |
+---+-----+-----+
1 rows in set (0.01 sec)
```

```
sqlite> select Description, Cost from parts where PartNo = 2619;
+-----+-----+
| Description      | Cost |
+-----+-----+
| Navigation Compass | 9.177 |
+-----+-----+
1 rows in set (0.01 sec)
```

```
sqlite>
```

SQL Server Console

SQL Server Console allows you to use a command-line interface. In other words, it provides interactive text-based screen for you to query input and result output from database.

Open Console

To open Console

- Open the connection and select **Tools** -> **Console** from the main menu or just simply press **Cmd-Shift-C**.
- Edit your SQL statement in the console command prompt.

Hint: To open a new console window you can also control-click the Connection/Database node of the navigation pane and select **Console** from the popup menu.

Hint: You are allowed to open multiple console windows which represents different connection each.

Exit Console

To exit Console

- Click the cross button at the main bar.

Example of Using SQL Server Console

Basic SQL Server query statements

```
1> use AdventureWorks;
2> select DepartmentID, Name from HumanResources.Department where
   GroupName = 'Executive General and Administration';
3> go
```

```
+-----+-----+
| DepartmentID | Name                |
+-----+-----+
|          9 | Human Resources    |
|         10 | Finance            |
|         11 | Information Services |
|         14 | Facilities and Maintenance |
|         16 | Executive          |
+-----+-----+
```

(5 rows affected)

```
1>
```

Dump SQL File

Navicat allows you to backup your database/schema/table(s) using the **Dump SQL File** feature.

- Select the database/schema/table(s).
- Control-click and select **Dump SQL File...** from the popup menu. If you select database/schema, you can choose dump **Structure + Data** or **Structure only**.
- Save the sql file to your destination.

Execute SQL File

Navicat allows you to restore your database/execute SQL file using the **Execute SQL File** feature.

- Select the database/schema.
- Control-click and select **Execute SQL File...** from the popup menu.

Note: You can drag a .sql file to a database or a server in the connection tree. Navicat will popup the Execute SQL File.

Example:

```
[Execute Batch log]
Server: Localhost
Database: report
Batch file: /Users/siuha/Desktop/report_sample.sql
(2010-03-16 10:46:42) execution started
(2010-03-16 10:46:43) execution finished
went through 35 queries
0 errors encountered
```

Convert to SQLite 3 (Available only for SQLite)

Navicat allows you to convert a SQLite 2 database file into a SQLite 3 database file.

- Control-click the connection and select **Convert to SQLite3...** from the popup menu.
- Save the database file to your destination.

View Database/Schema/Table Structure (Available only in Full Version)

Navicat allows you to view and print database, schema and table structure. Just simply control-click the database/schema and select **View Tables Structure -> In HTML Format** or control-click the table(s) and select **View Tables Structure**.

Log Files

Navicat provides **Log Files**, namely **QueryExec.log** and **CmdLineJob.log**, to keep track on the actions that have been performed in Navicat and they are located in a default folder, e.g. `~/Library/Application Support/PremiumSoft CyberTech/Navicat Premium`. You are allowed to change the log files location under **Preferences**.

- **QueryExec.log**
Stores all SQL statements of all the operations executed over databases and database objects in Navicat.

Hint: Simply click View -> Query Log or press Cmd-L to open the QueryExec.log file.

- **CmdLineJob.log**
Stores information for Navicat command line process and all operations while running schedule.