CUBRID Query Browser User v1.1.0 Manual

Contents

CUBRID Query Browser

This chapter explains how to use the CUBRID Query Browser, a GUI-based query tool. The CUBRID Query Browser facilitates various tasks allowing users to execute SQL statements against the connected database.

CUBRID Query Browser consists of a database server running on a host where the Broker is installed, and a GUI client. The CUBRID Query Browser Client utility is written in Java and can be run in any environment that supports Java.

This chapter covers the following topics:

- Introduction to the CUBRID Query Browser
- Install and Running CUBRID Query Browser
- Client Features of the CUBRID Query Browser

CUBRID Query Browser Introduction

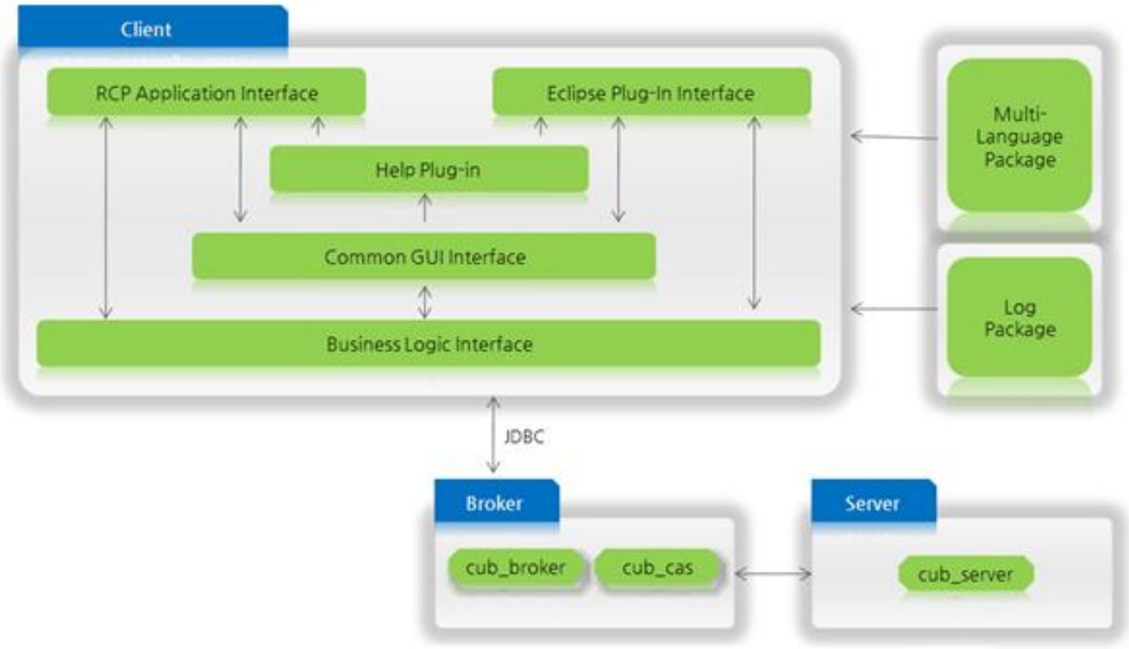
The CUBRID Query Browser is a tool what users can use to execute SQLs and manage the objects of CUBRID databases conveniently.

It can run as a RCP application or run as a plug-in in eclipse IDE.

It replaced the Query Editor of CUBRID Manager before CUBRID Manager 8.3.1 and is integrated into CUBRID Manager as a plug-in after CUBRID Manager 8.3.2

CUBRID Query Browser Architecture

The figure below shows the architecture of the CUBRID Query Browser. It is developed as an eclipse RCP application and eclipse plug-in. And as the figure shows, it visits the CUBRID database server through JDBC driver and CUBRID broker server.



Getting started

How to get the CUBRID Query Browser

Requirements

- **JDK 1.6 or higher** version is required.
- When CUBRID Query Browser is running as a plug-in in eclipse IDE, **3.5 or higher** version of eclipse IDE is recommended.
- If the operation system is a 32-bit system, the 32-bit JDK is required and the CUBRID Query Browser must be 32-bit version. And the 64-bit system needs 64-bit JDK and 64-bit CUBRID Query Browser installation file.

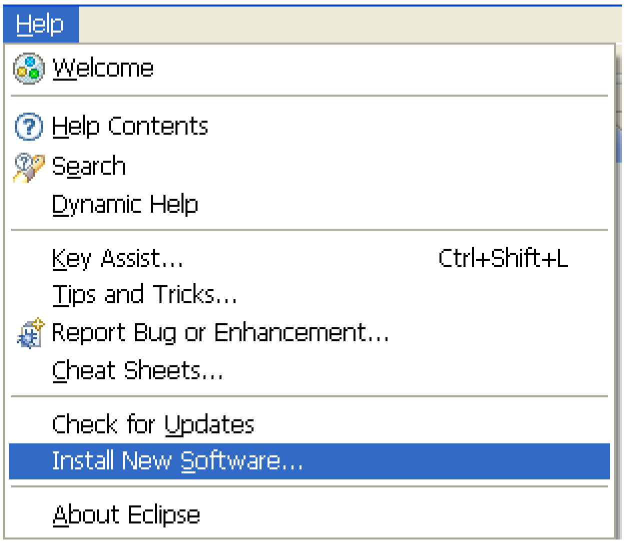
Download CUBRID Query Browser RCP Application

Please visit the web site https://sourceforge.net/projects/cubridtools/ for more information of CUBRID Query Browser or other CUBRID tools.

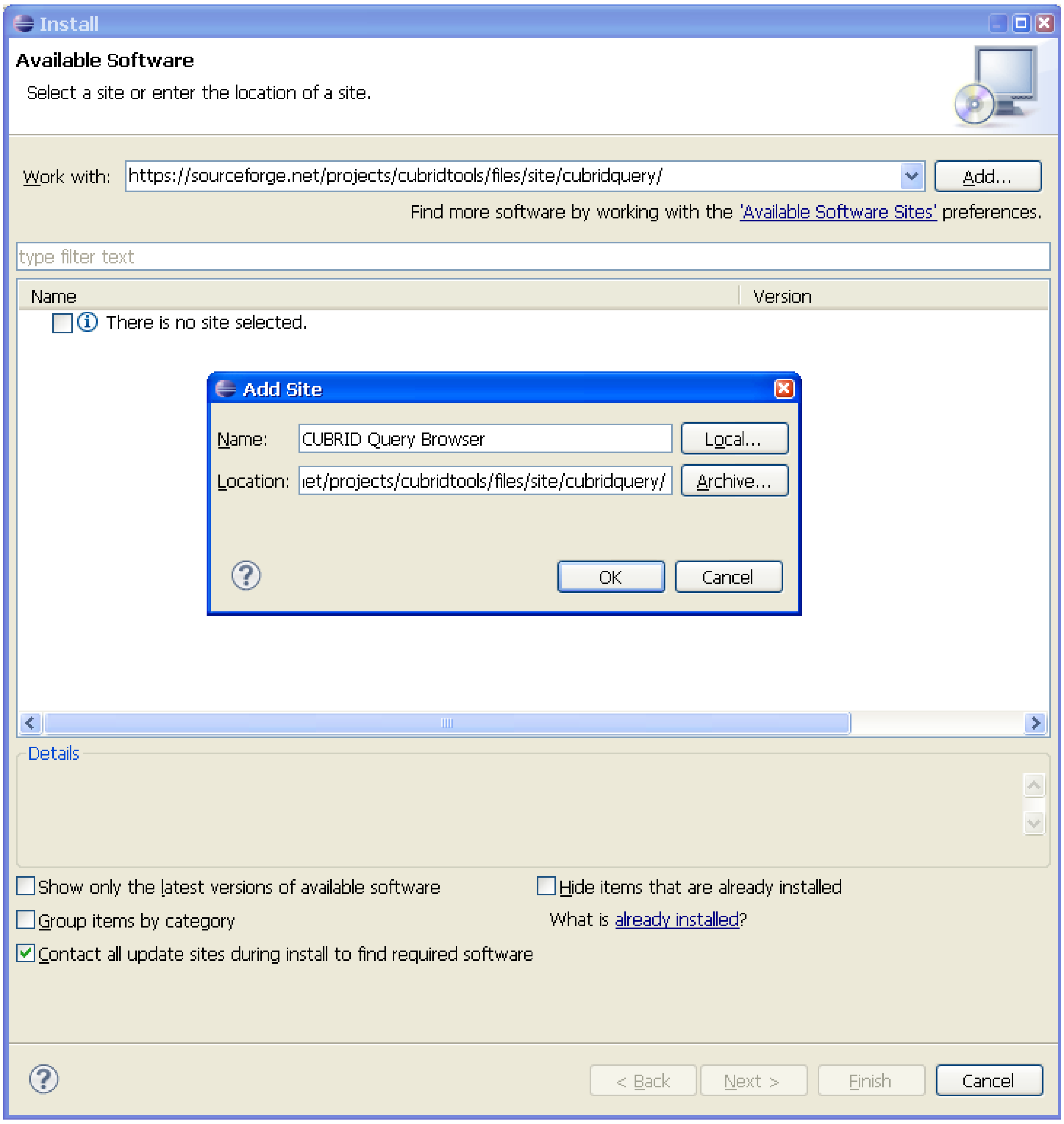
Install CUBRID Query Browser plug-in into Eclipse IDE

The steps of how to install it in eclipse IDE are listed:

- Step 1: Open [Help]→[Install New Software]



- Step 2: After 'Install New Software' dialog opened, use [Add...] to add the URL http://sourceforge.net/projects/cubridtools/files/site/cubridquery/ to 'Available Software Sites'.



Install

Available Software

Select a site or enter the location of a site.

Work with: https://sourceforge.net/projects/cubridtools/files/site/cubridquery/

Find more software by working with the 'Available Software Sites' preferences.

type filter text

Name	Version
□ ⓘ There is no site selected.	

Add Site

Name: CUBRID Query Browser

Location: ıet/projects/cubridtools/files/site/cubridquery/

Details

□ Show only the latest versions of available software

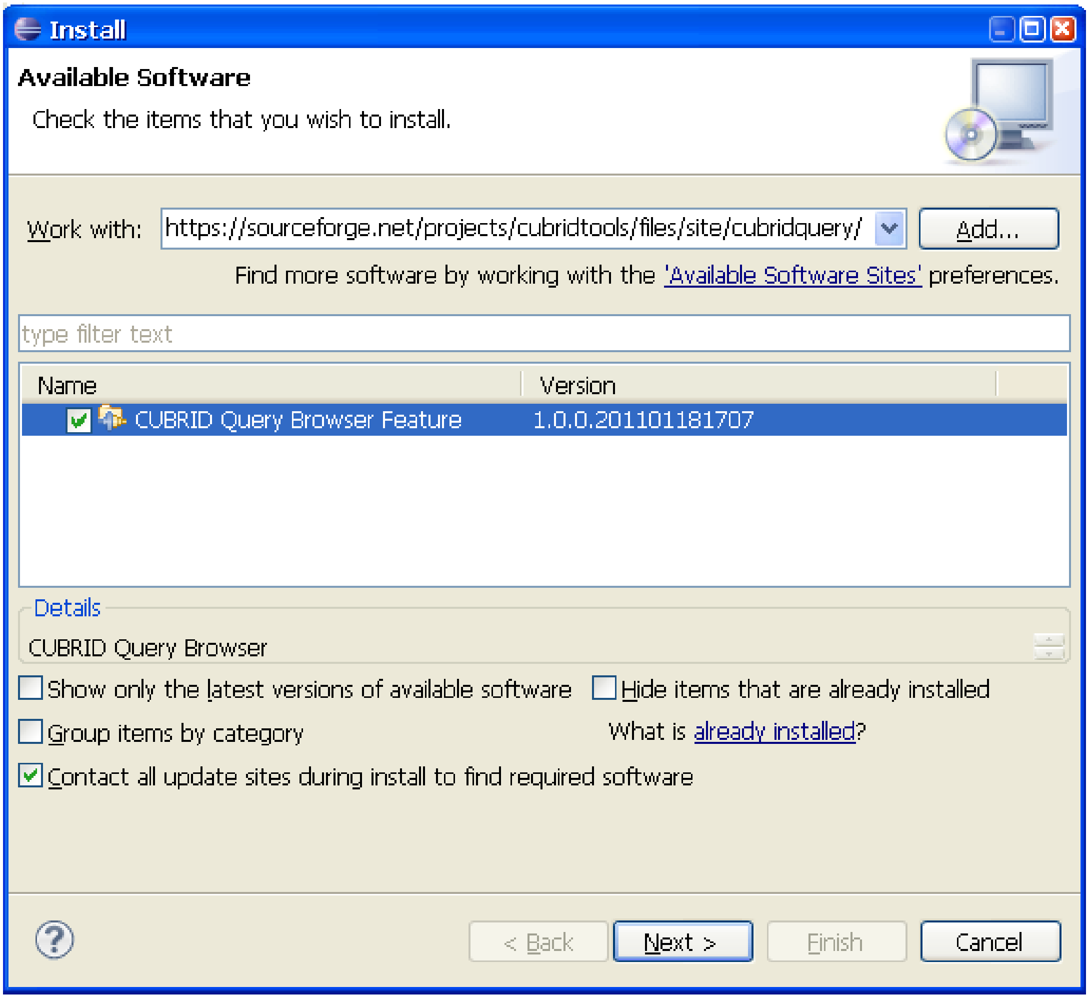
□ Hide items that are already installed

□ Group items by category

What is already installed?

☑ Contact all update sites during install to find required software

- Step 3: Check the last version of the CUBRID Query Browser and use the wizard to finish the installation.



Available Software

Check the items that you wish to install.

Work with: https://sourceforge.net/projects/cubridtools/files/site/cubridquery/

Find more software by working with the 'Available Software Sites' preferences.

type filter text

Name	Version
☑ CUBRID Query Browser Feature	1.0.0.201101181707

Details

CUBRID Query Browser

□ Show only the latest versions of available software

□ Hide items that are already installed

□ Group items by category

What is already installed?

☑ Contact all update sites during install to find required software

After installation, restarting the eclipse is strongly recommended.

How to run CUBRID Query Browser

Requirements

- JDK **1.6 or higher** is required.
- When CUBRID Query Browser is running as a plug-in in eclipse IDE, **3.5 or higher** version of eclipse IDE is recommended.
- If the operation system is a 32-bit system, the 32-bit JDK is required and the CUBRID Query Browser must be 32-bit version. And the 64-bit system needs 64-bit JDK and 64-bit CUBRID Query Browser.

Run as a RCP application

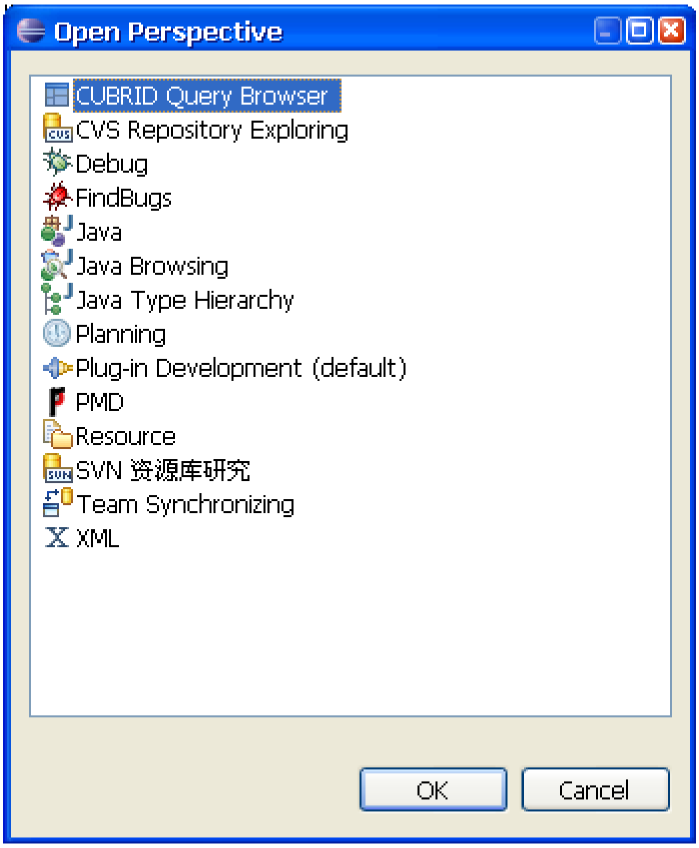
- In **LINUX**: Extract all files of the downloaded file into a dictionary and execute 'cubridquery'.
- In **MAC**: Extract all files of the downloaded file into a dictionary and execute 'cubridquery'.
- In **WINDOWS**: Extract all files of the downloaded file into a dictionary and execute 'cubridquery.exe'.

Run in CUBRID Manager

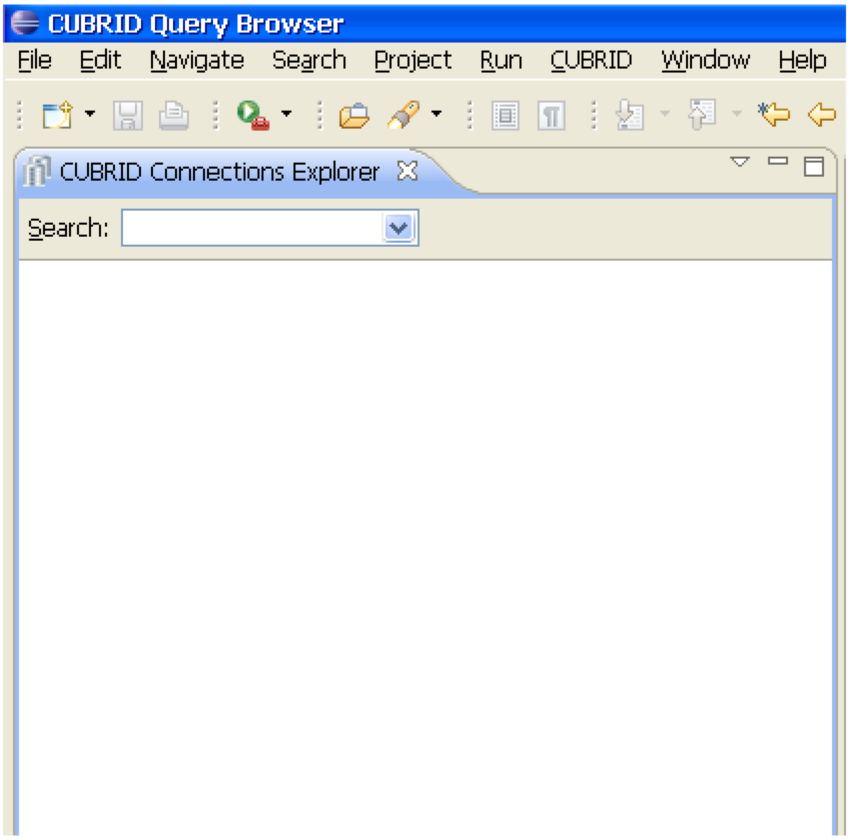
It is integrated into CUBRID Manager as a plug-in after CUBRID Manager 8.3.2. CUBRID Manager 8.4.0 include the CUBRID Query Browser all functions before 1.1

Run in eclipse IDE

After the installation is finished and the eclipse is restarted, a new item named 'CUBRID Query Browser' will be in the [Open Perspective] dialog as follows.



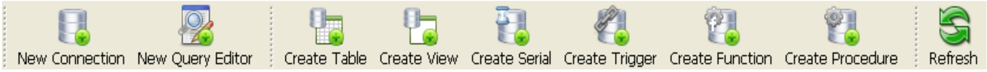
Select it and click [OK], the view of eclipse will be as follows and now you can start to use CUBRID Query Browser.


CUBRID Query Browser
File Edit Navigate Search Project Run CUBRID Window Help
CUBRID Connections Explorer
Search:

Concepts

Main Toolbar

These buttons of tool bar are some shortcuts of the main menu items of CUBRID Query Browser.



Option	Icon	Shortcut Key	Description
New Connection			Create a new CUBRID Database Connection when it is clicked.
New Query Editor			Create a new query editor window of current selected connection when it is clicked.
Create Table			Open Create Table Dialog when it is clicked.
Create View			Open Create View Dialog when it is clicked.
Create Serial			Open Create Serial Dialog when it is clicked.
Create Trigger			Open Create Trigger Dialog when it is clicked.
Create Function			Open Create Function Dialog when it is clicked.
Create Procedure			Open Create Procedure Dialog when it is clicked.
Refresh			Refresh current node which is selected when it is clicked.

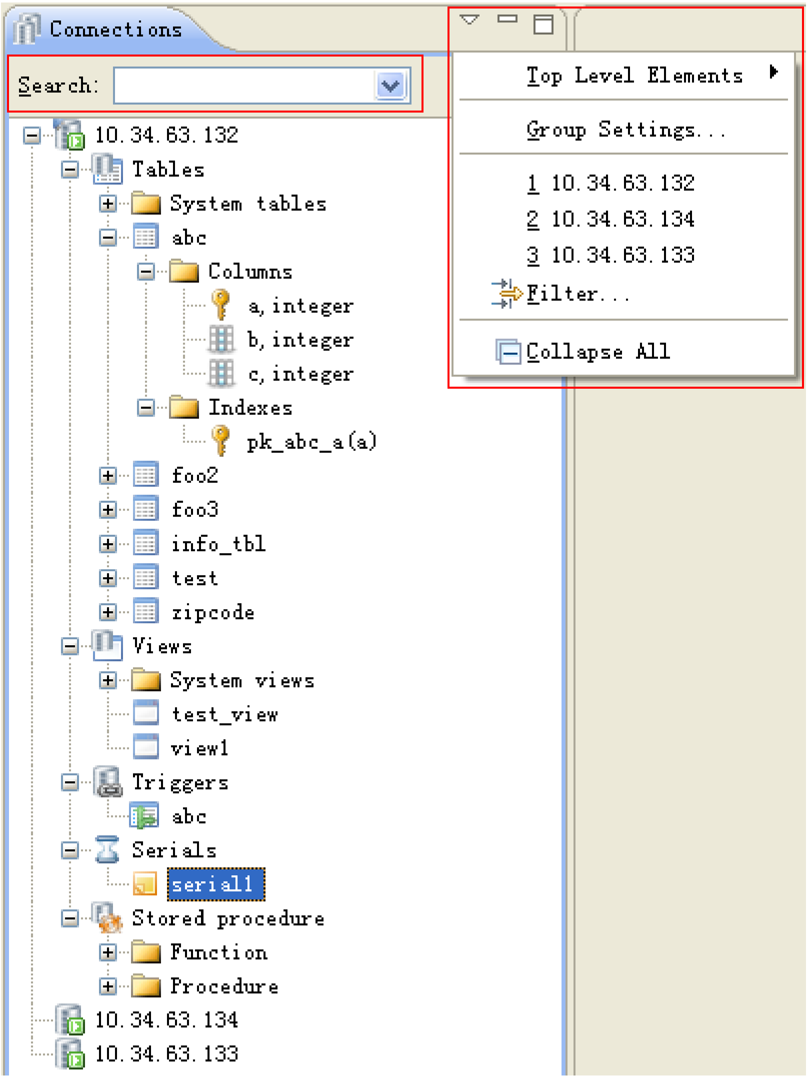
Connection Explorer

Connection Explorer

The left view which title is "Connections" is a database connection explorer. Each Database Connection Node in it is a container that stores a set of parameters which can be used to connect a CUBRID database through CUBRID JDBC driver.

And when the connection is opened, it can list the database's objects such as tables, columns and indexes of tables, views, triggers, serials and procedures with a tree view structure. Users can do something with nodes that are listed such as creating/dropping tables; creating/dropping views and etc by execute its context menus popped by right clicked.

Connection Explorer have it's menubar and toolbar.



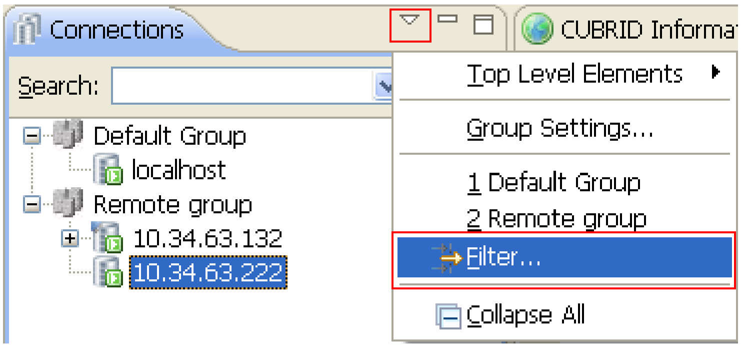
Object Find and Filter

Find Object

The text editor named “Search” in connection explorer toolbar is used to find locate the node of the tree by node’s name. It support the pattern, * stand for any string, ? stand for any character.

Filter Object

You can filter the elements in connection explorer by the below ways.

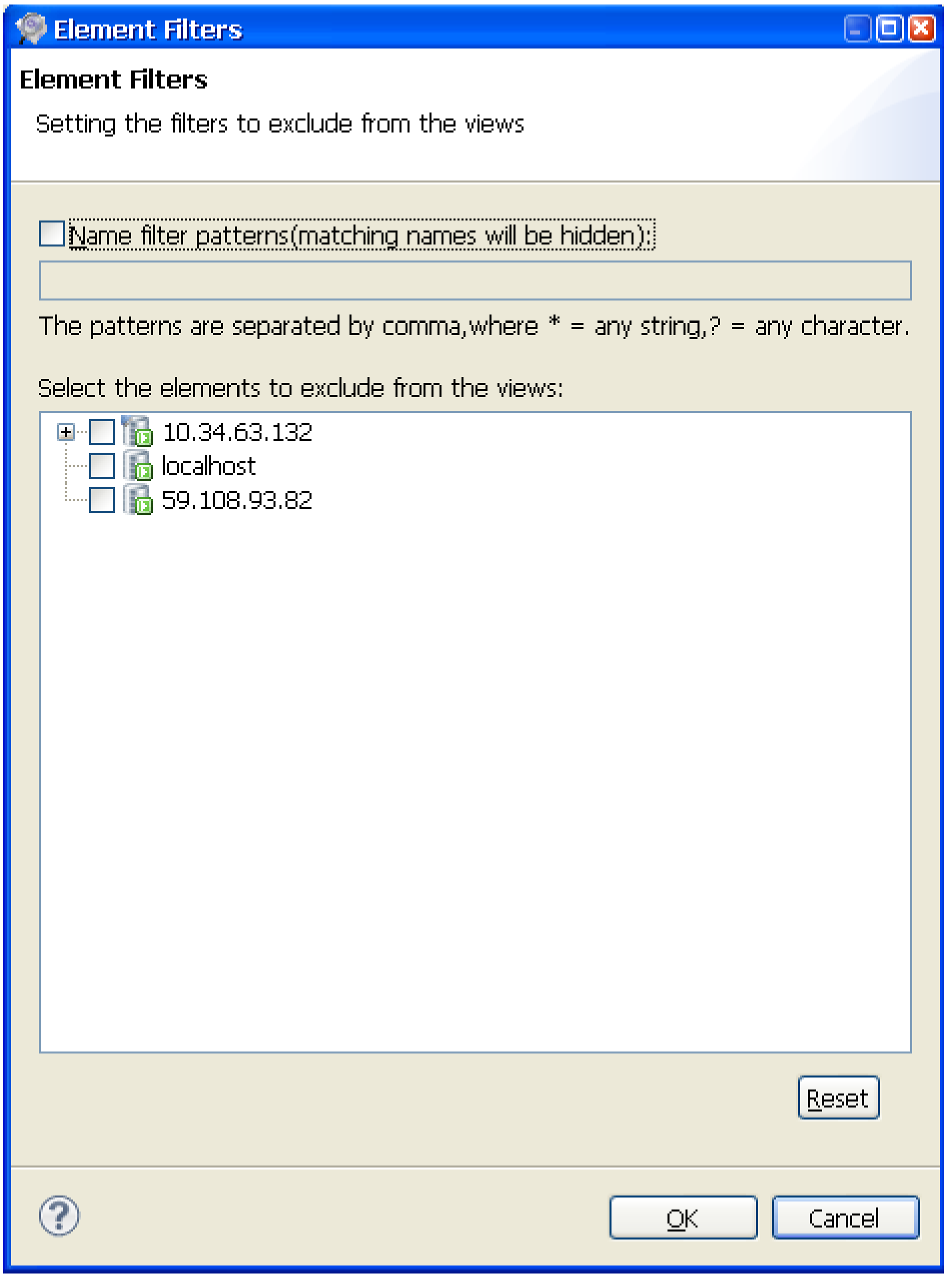


The menus will be popped when the button which’s shape is triangle is left clicked.

(1) Use checkable action to show all top level elements in navigator menu. When checked, hide this element; otherwise, show this element.

(2) By Filter dialog

If the menu named “Filter…” is clicked, a filter setting up dialog of the tree view will be opened as follows:



Element Filters

Setting the filters to exclude from the views

□ Name filter patterns(matching names will be hidden):

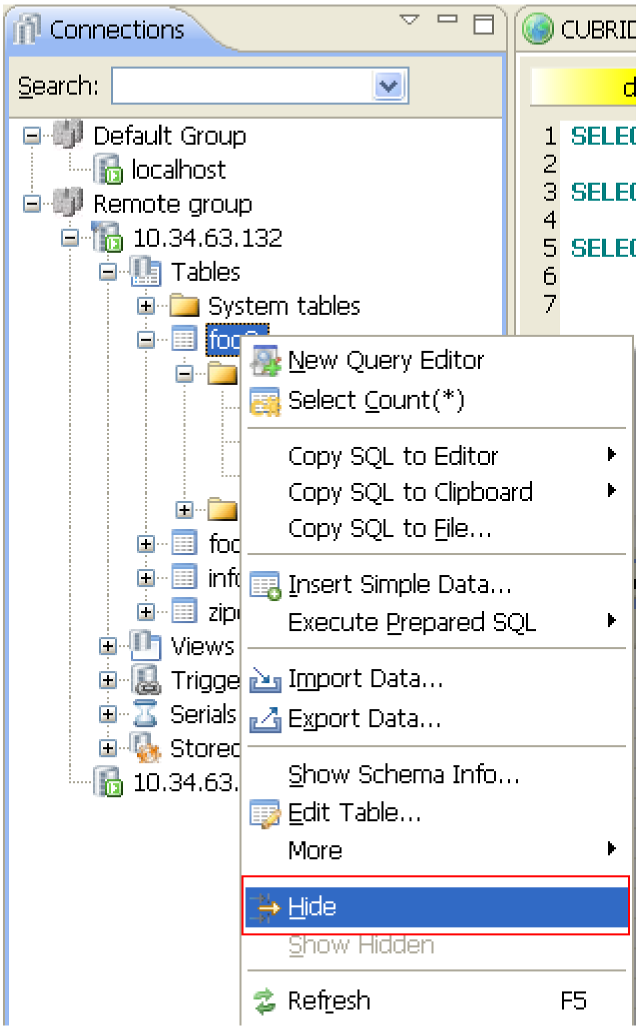
The patterns are separated by comma,where * = any string,? = any character.

Select the elements to exclude from the views:

- □ 10.34.63.132
- □ localhost
- □ 59.108.93.82

The nodes which are checked in the tree view of dialog will not displayed in the Database connection node tree view. Or if the “name filter patterns” is setting with expression, the nodes which have the names that match the patterns will be hided. The patterns can be split by “,”. For example “test*” pattern will make all nodes which name is started with “test” hided such as “test-db”, ”test1234”…, and the “demo?” pattern will hide the nodes which name is started with “demo” and the length of name is 5, such as “demo1”, “demo2”…

(3) All nodes in the connection explorer have two context menus [Hide] and [Show Hiden].



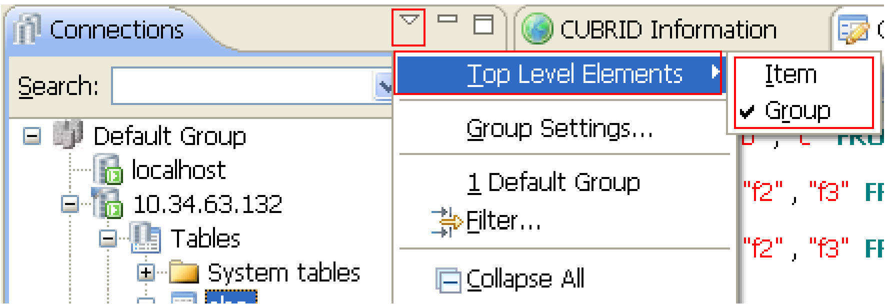
[Hide] will hide current selected node and all children of it.

[Show Hidden] will make all children of the selected node visible.

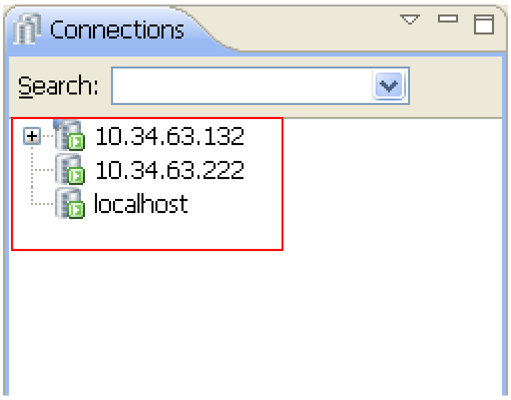
Connection Group

If there are lots of database connections in the connection explorer, users can use group manager to classify those connections.

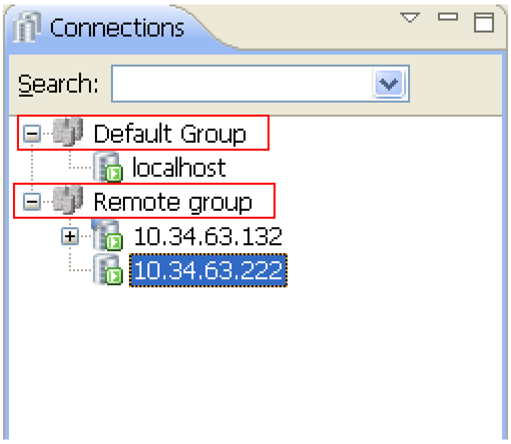
The explorer supports two kinds of view style to display connections-[Item] or [Group].



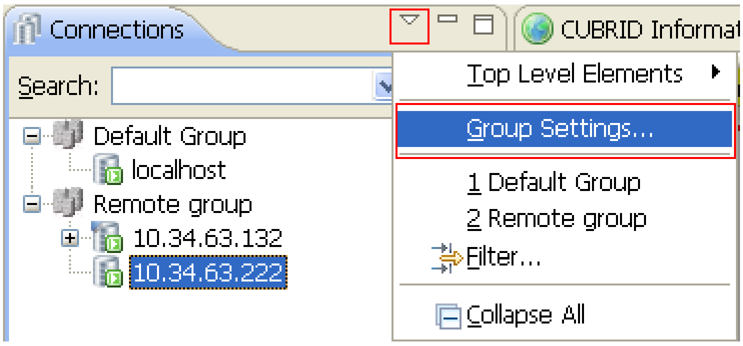
If item style is selected, all connections will be listed in the explorer without grouped. As follows:



If group style is selected, connections will be move to specific groups. As follows:

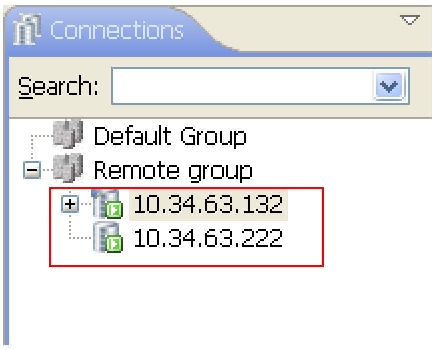


And users can use **[Group settings]** to manage the groups. The menu is as follows:



Database Connection

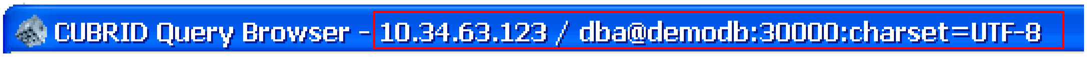
Database connections are listed in the connection explorer and they store the information of how to connect a specific CUBRID database server.



Users can add, edit and delete the database connections by using menus.

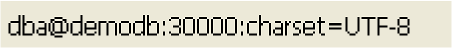
Application Title and Status

In the title of application, current database connection information will be displayed as follows:



CUBRID Query Browser - 10.34.63.123 / dba@demodb:30000:charset=UTF-8

And in the status bar it will be shown as follows:



dba@demodb:30000:charset=UTF-8

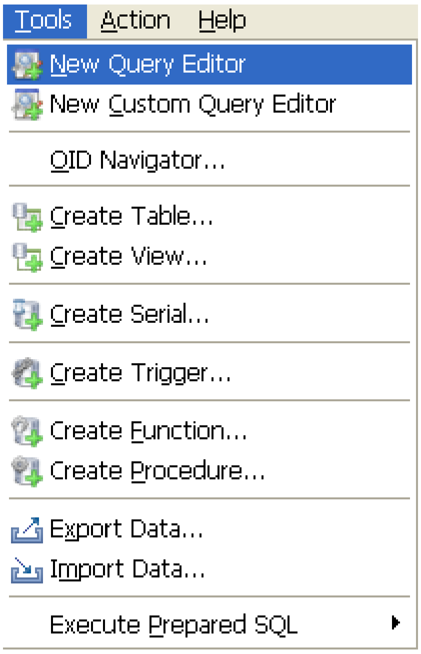
And the information will be changed with the changing of the selection of connection node or query editor.

Query Editor

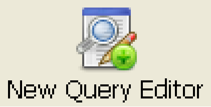
Open Query Editor

There are lots of ways to open a query editor.

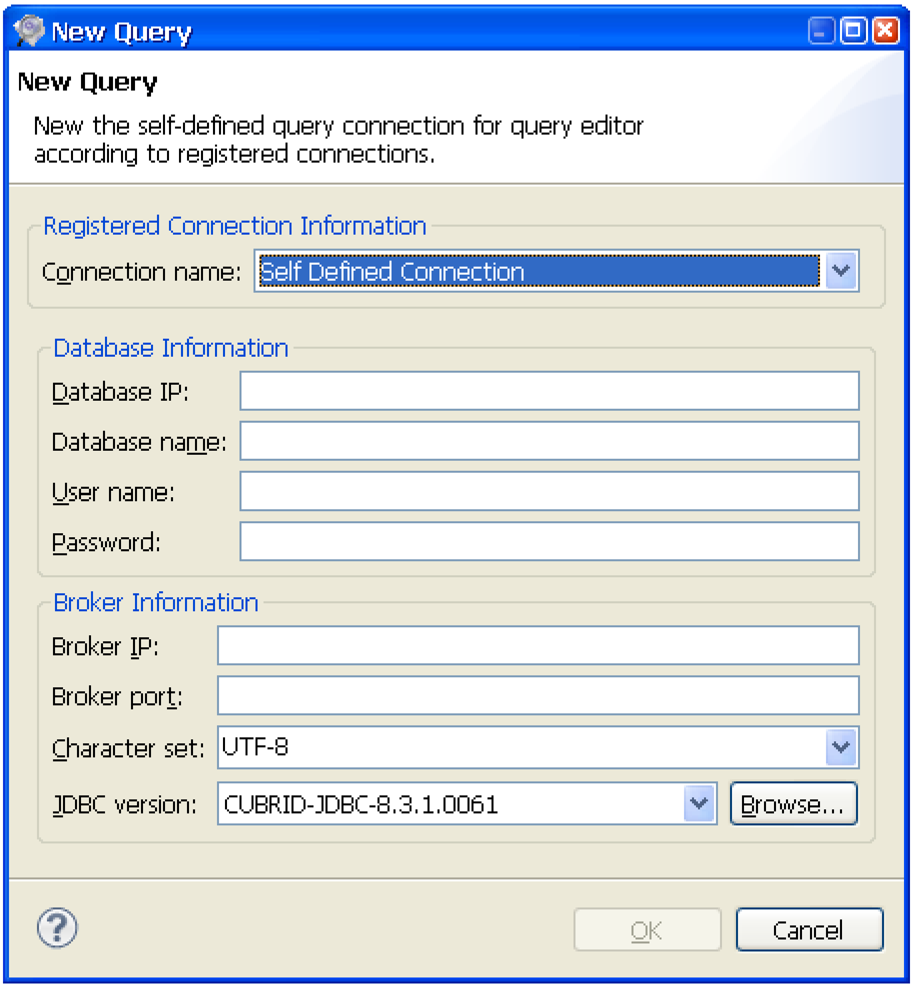
- If there is no database connection selected in the tree view, the menu [Tools]→[New Query Editor] is as same as the button [New Query Editor] in the tool-bar. Both of them will open a new Query Editor and users should input the database connection information into the dialog opened.



or



The dialog will be opened as follows:



New Query

New Query

New the self-defined query connection for query editor according to registered connections.

Registered Connection Information

Connection name: Self Defined Connection

Database Information

Database IP:

Database name:

User name:

Password:

Broker Information

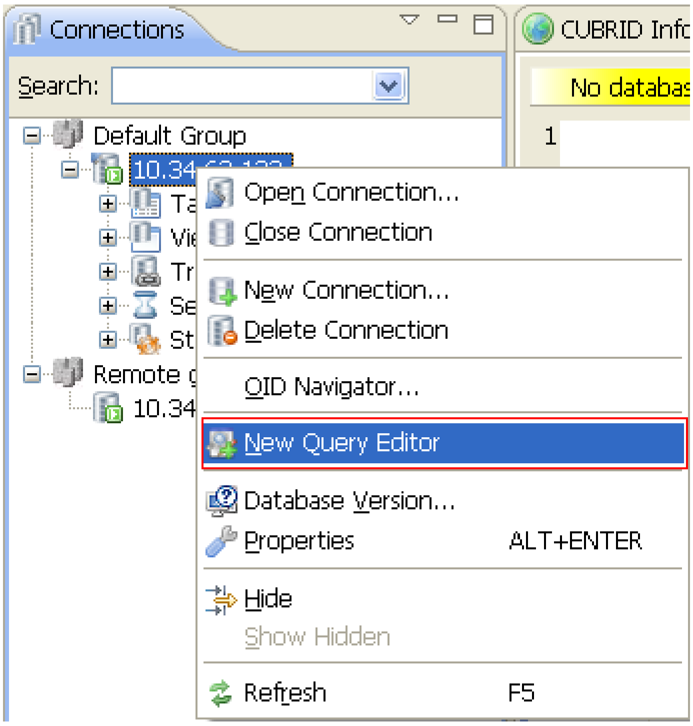
Broker IP:

Broker port:

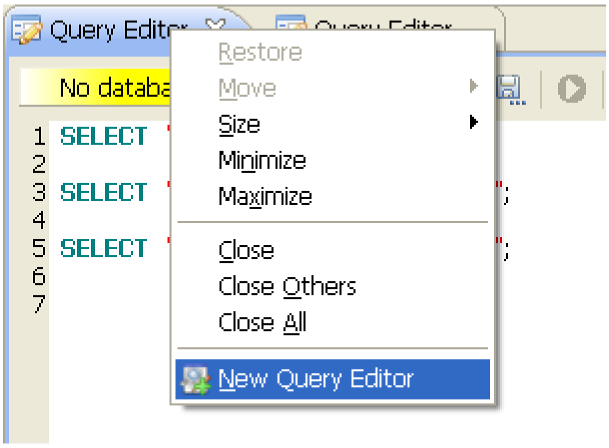
Character set: UTF-8

JDBC version: CUBRID-JDBC-8.3.1.0061

- The context menu **[Open Query Editor]** of database connection/table/column/view will open a new query editor.

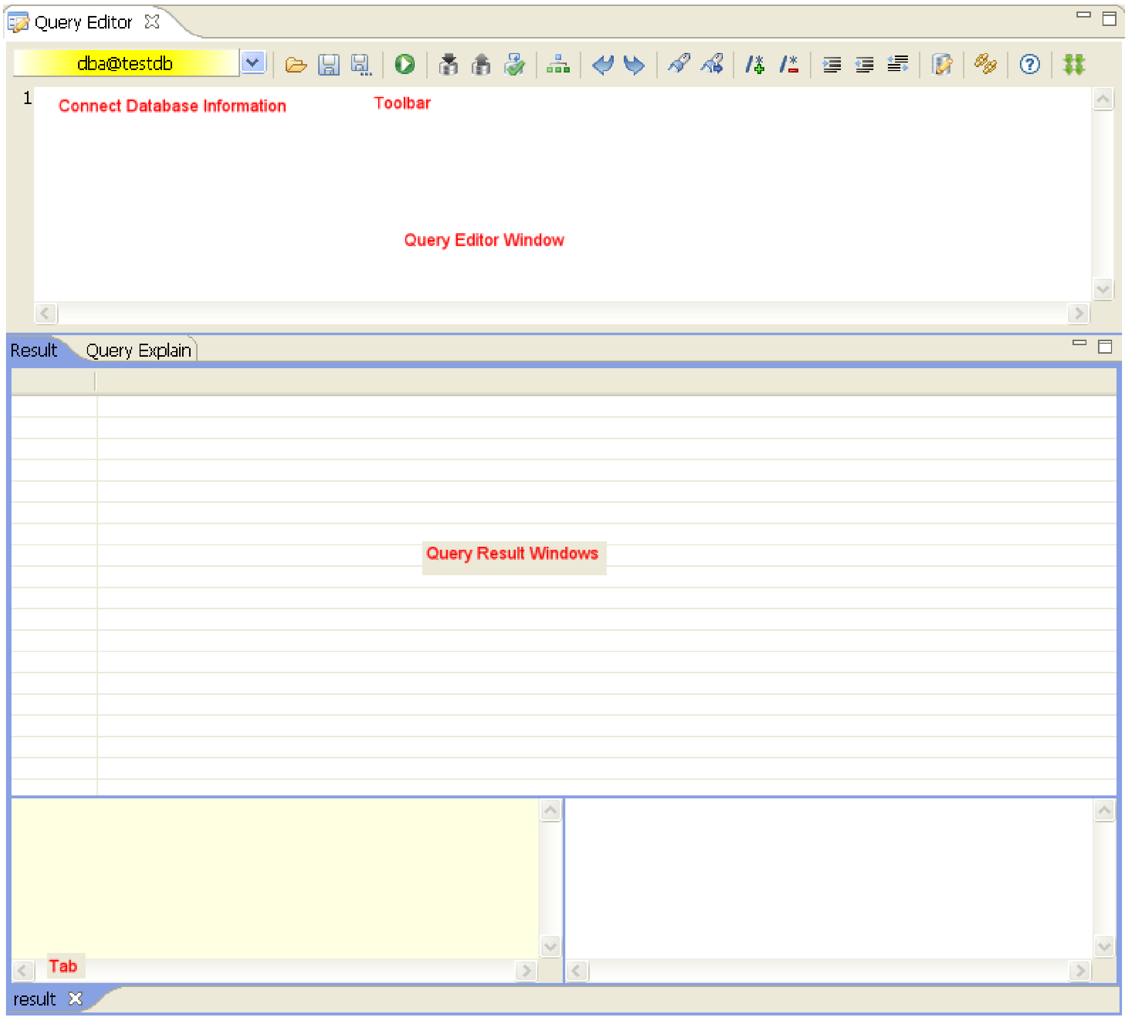


- The context menu of query editor tab also has an item to open a new query editor with current query editor's database connection:



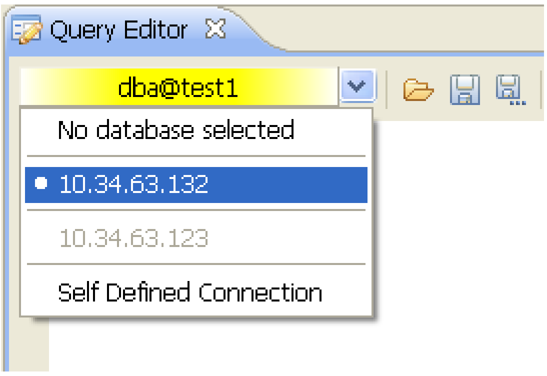
Query Editor Overview

Users can do a lot of works such as edit/execute SQLs, check results of the executions, view logs, check execution plan and etc by using Query Editor.



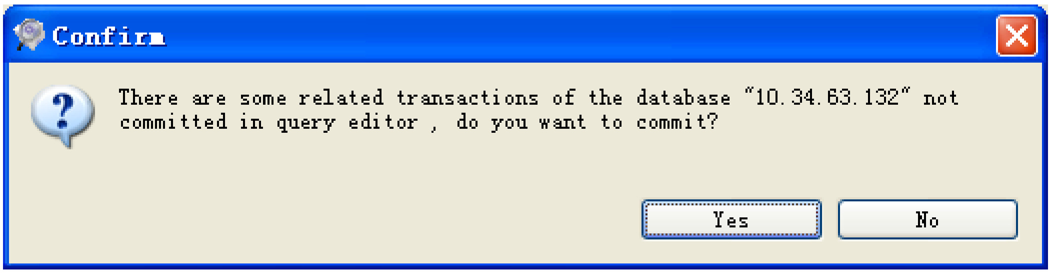
Query Editor Toolbar

Users can change the database connection of current Query Editor by changing this option.



The yellow highlighted sections in the figures above indicate that current database connection is named "10.34.63.132" and the login user is named DBA and the login database is "test1".

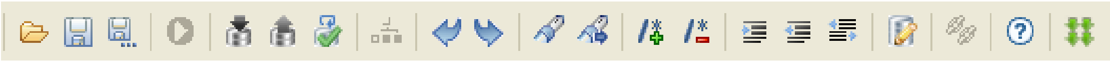
If a different database connection is selected when the query editor has a transaction not finished, a dialog will be popped to prompt how to deal with the not finished transaction: commit or rollback or cancel current changing:



Confirm

There are some related transactions of the database "10.34.63.132" not committed in query editor , do you want to commit?

The edit function of the Query Editor is synchronized with **[Edit]** on the main menu. The following table shows the options available on the Query Editor's toolbar: Major options are provided with a shortcut key.



Option	Icon	Shortcut Key	Description
Open	📂		Opens a text-based SQL and displays it in the query edit panel.
Save	💾		Saves the contents of the query edit panel.
Save As	💾		Saves the contents of the query edit panel in a different name.
Run	▶	F5	Executes all the queries in the query edit panel. Alternatively, executes queries selected as a block. This icon is deactivated when the execution is unavailable (▶).
Commit	⬇		Keeps deactivated in auto-commit mode (⬇); while in non-auto-commit mode, allows you to commit (⬇) explicitly when a transaction occurs.
Rollback	⬆		Keeps deactivated in auto-commit mode (⬆); while in non-auto-commit mode, allows you to rollback (⬆) explicitly when a

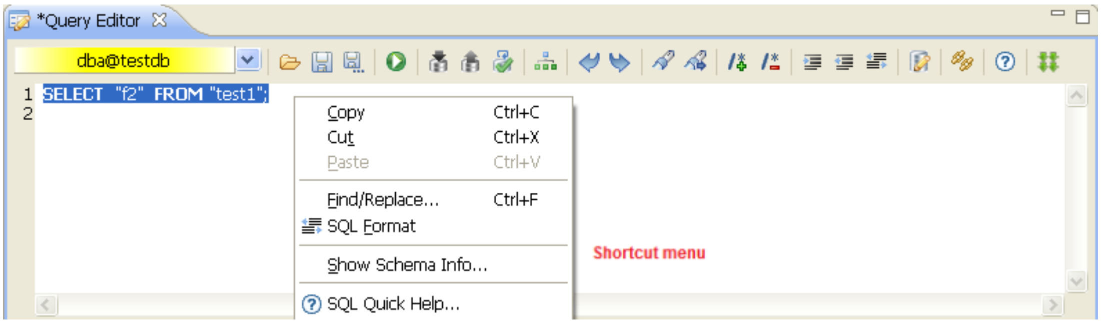
			transaction occurs.
auto commit			Automatically commits queries executed in the query edit panel. You can toggle the icon. When auto commit is on, the icon is displayed as , and when it is off, the icon is displayed as . Note that the setting is only applied to the corresponding Query Editor; that is the default value of the option is not changed.
Display query plan		F6	Displays the execution plan of the selected query. For more information, see [Query Execution Plan]().
Undo	↶	Ctrl+Z	Cancels the most recent edit action.
Redo	↷	Ctrl+Y	Re-executes the most recent edit action that is canceled by Undo.
Find/Replace		Ctrl+F	Provides search and replace function that can be used in the query edit panel.
Find next		F3	Provides further search for items that are already searched once.
Convert to comment	/*	Ctrl+/	Adds the comments to the selection area or to the line where the cursor is located in the query edit panel. Comment strings are inserted with a double dash (--).
Delete comment string	/*	Ctrl+/	Deletes the comments from the selection area or from the line where the cursor is located in the query edit panel.
Insert tab		Tab	Indents the selection area in the query edit panel.
Delete tab		Shift+Tab	Un-indents the selection area in the query edit panel.

Format SQL		Ctrl+Shift+F	Formats the selected SQL strings in the query edit panel.
Get OID info			A toggle button available on the toolbar in the query edit panel for [Get OID info], which can be found in the Query Editor options. That is, by selecting this button on the toolbar when [Get OID info], which can be found in the Query Editor options, is OFF, you can directly modify/delete data in the query results panel for subsequent queries. In this button is selected on the toolbar, however, this only applies to the corresponding Query Editor. The values of the Query Editor options for the corresponding database won't be changed.
Set defined SQL parameter			To specify a parameter whenever the statement is executed, write Prepared Statement, select the written statement, and click the icon.
SQL Quick Help	?		Find the help about the words which you selected in the SQL edit panel.
Show or Hide Query result panel			Show or Hide query result panel.

Edit SQL

Shortcut Menu

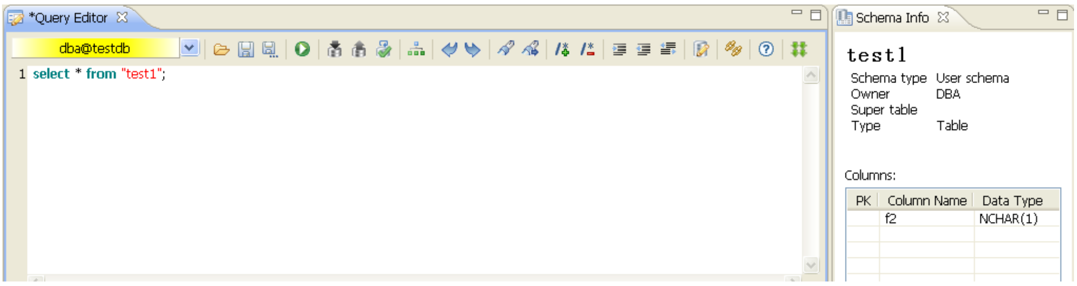
If you right-click on the query edit panel, you can select Copy, Cut, Paste, Find/Replace, SQL Format and Show Schema Info. The editing function is synchronized with [Edit] of the menu.



```
SELECT "f2" FROM "test1";
```

Show Schema Info

If you select a table name and then execute [Show Schema Info], you can create a query while seeing the Query Editor view and the Schema Info view on the same window. Note that you can view the schema information only when executing Query Editor in the Explorer (toolbar not supported).



```
select * from "test1";
```

test1

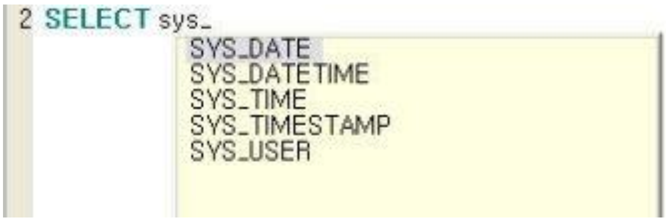
Schema type User schema
Owner DBA
Super table
Type Table

Columns:

PK	Column Name	Data Type
	f2	NCHAR(1)

Automatic Completion

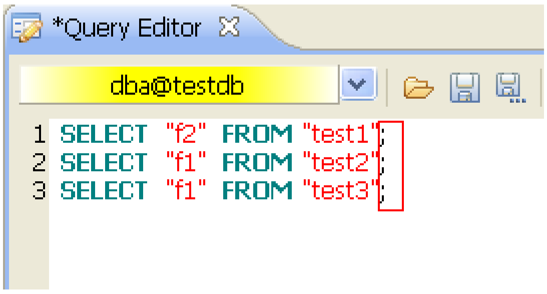
If you enter a CUBRID database keyword, CUBRID automatically finds and completes the statement, increasing usability and user accessibility.



```
SELECT sys_
```

Executing Multiple SQLs

Enter a semicolon at the end of the query statement to specify the end of one query and the start of the next. If there are multiple queries, they are executed sequentially. Each query creates a corresponding tab in the query results panel. If you execute multiple queries without separating them with semi-colons, only the first query is executed, with the rest ignored.



```
SELECT "f2" FROM "test1";
SELECT "f1" FROM "test2";
SELECT "f1" FROM "test3";
```

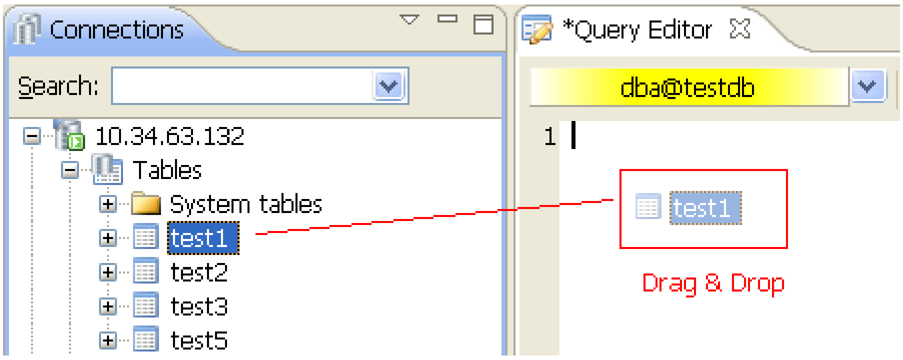
Executing SQL Quickly

Press Ctrl+Enter will execute the sql where current cursor is.

Drag and Drop

If you drag and drop a table or tables to retrieve from the connection navigation tree into the query edit panel of the Query Editor, a **SELECT** query statement is created automatically. Multi-tables drag and drop is supported.

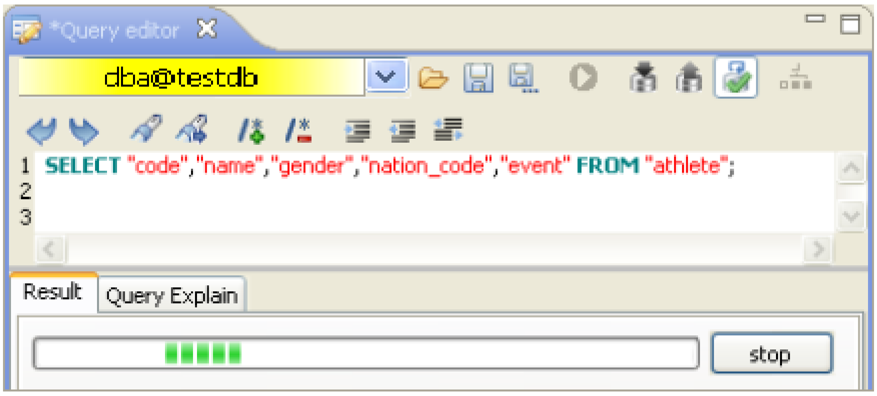
And the columns of tables also support drag and drop to the query editor.



Cancel SQL Executions

It is a function to stop current running SQL execution. This can be divided into two functions.

- **When executing multiple queries**：If you click the Stop button when multiple queries are being executed with the auto commit button enabled, queries processed before the stop operation are reflected normally, and the currently canceled query and following ones are not reflected. If the auto commit button is not enabled, no executed queries are reflected.
- **When executing a long time transaction**: If you click the Stop button when executing a long time transaction, a query stop command is delivered to CUBRID Manager > JDBC > Broker > Server, and the query finally stops in the database server the query is actually canceled. For Windows, this function is supported in CUBRID 2008 R2.2 or later.



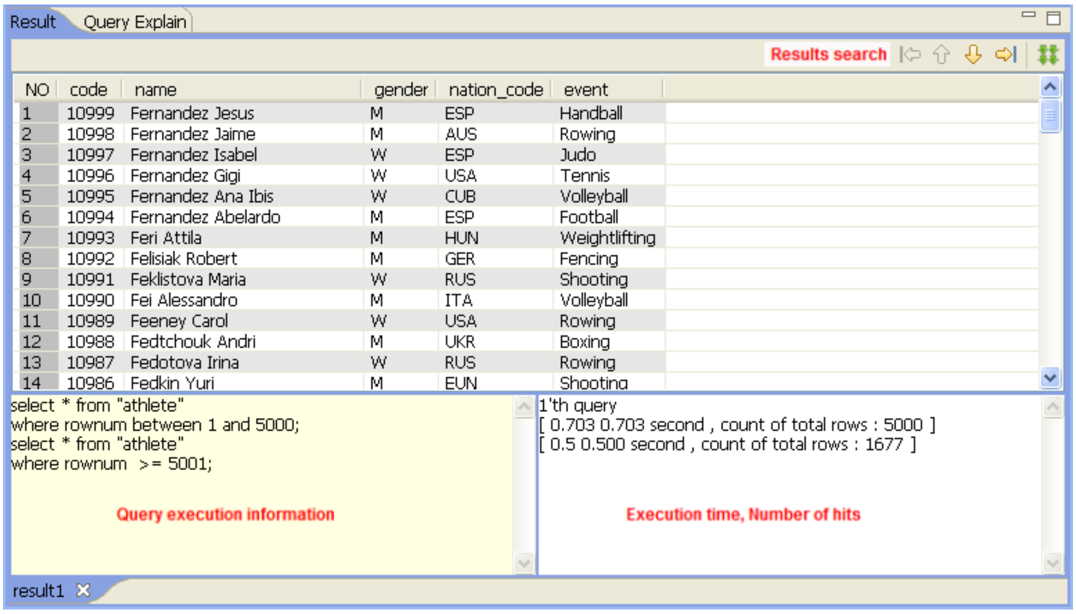
```
SELECT "code","name","gender","nation_code","event" FROM "athlete";
```

SQL Execution Results Panel

Structure

The query results panel displays the results of the executed query. If there are multiple queries executed, the results of each query are displayed in separate tabs. Users can check the query results by selecting the corresponding tab.

The query results panel is divided into areas where you can navigate the results, view information of the executed query, and check execution time and the number of results returned by the query.



NO	code	name	gender	nation_code	event
1	10999	Fernandez Jesus	M	ESP	Handball
2	10998	Fernandez Jaime	M	AUS	Rowing
3	10997	Fernandez Isabel	W	ESP	Judo
4	10996	Fernandez Gigi	W	USA	Tennis
5	10995	Fernandez Ana Ibis	W	CUB	Volleyball
6	10994	Fernandez Abelardo	M	ESP	Football
7	10993	Feri Attila	M	HUN	Weightlifting
8	10992	Felisiak Robert	M	GER	Fencing
9	10991	Feklistova Maria	W	RUS	Shooting
10	10990	Fei Alessandro	M	ITA	Volleyball
11	10989	Feeney Carol	W	USA	Rowing
12	10988	Fedtchouk Andri	M	UKR	Boxing
13	10987	Fedotova Irina	W	RUS	Rowing
14	10986	Fedkin Yuri	M	EUN	Shooting

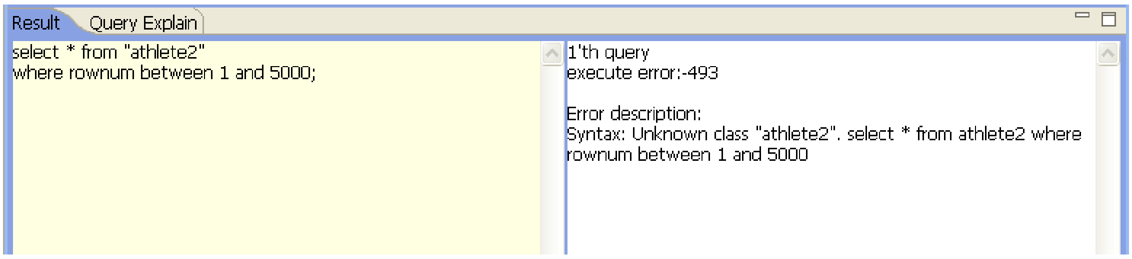
```
select * from "athlete"
where rownum between 1 and 5000;
select * from "athlete"
where rownum  >= 5001;
```

Query execution information

```
1'th query
[ 0.703 0.703 second , count of total rows : 5000 ]
[ 0.5 0.500 second , count of total rows : 1677 ]
```

Execution time, Number of hits

If a SQL statement such as **SELECT** is executed or if there is an error when the **SQL** statement executed, query execution information and error message or execution time information are displayed in the Logs tab.



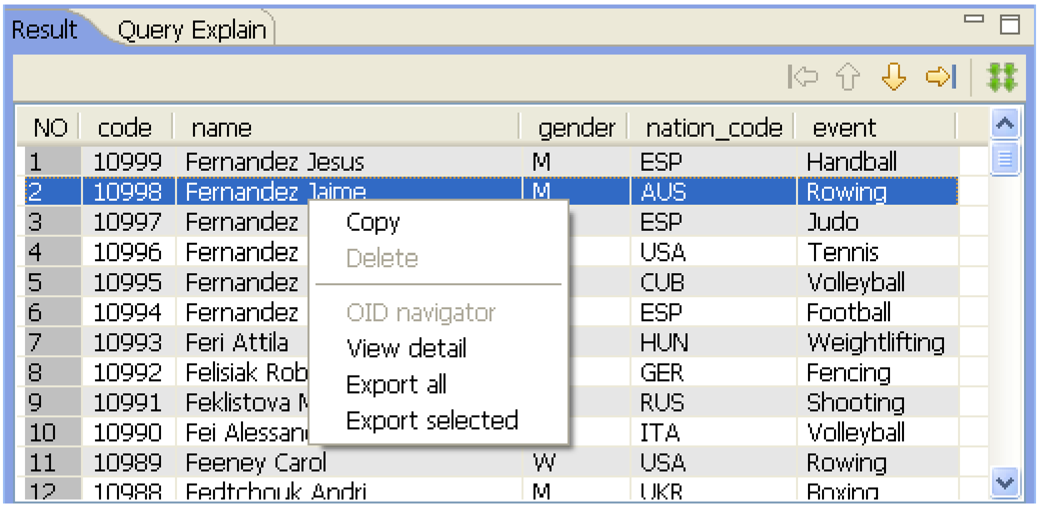
```
select * from "athlete2"
where rownum between 1 and 5000;
```

```
1'th query
execute error:-493

Error description:
Syntax: Unknown class "athlete2". select * from athlete2 where
rownum between 1 and 5000
```

- **Results search**: You can navigate the entire search results while moving by the value set in [Page unit of result instances] of **[Query options]**.
- **Query execution information**: Shows from which query the current result comes.
- **Execution time and Number of hits**: Provides information about the execution time on the server to get the current query results and total search count.

Context Menus

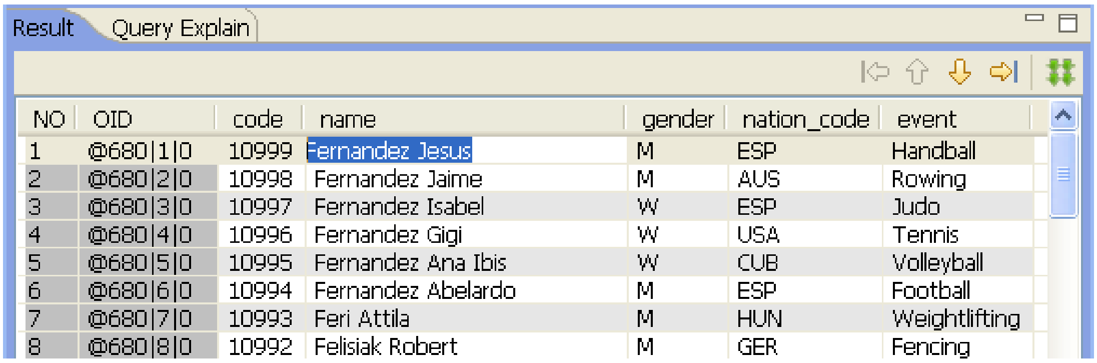
As shown in the figure below, the query results panel's shortcut menu contains some options such as Copy, Delete, OID navigator, View detail, Export all and Export selected.



NO	code	name	gender	nation_code	event
1	10999	Fernandez Jesus	M	ESP	Handball
2	10998	Fernandez Jaime	M	AUS	Rowing
3	10997	Fernandez		ESP	Judo
4	10996	Fernandez		USA	Tennis
5	10995	Fernandez		CUB	Volleyball
6	10994	Fernandez		ESP	Football
7	10993	Feri Attila		HUN	Weightlifting
8	10992	Felisiak Rob		GER	Fencing
9	10991	Feklistova M		RUS	Shooting
10	10990	Fei Alessan		ITA	Volleyball
11	10989	Feeney Carol	W	USA	Rowing
12	10988	Fedtchouk Andri	M	UKR	Boxing

Copy: Copies an entire row. To copy a specific column of the row only, select [View detail].

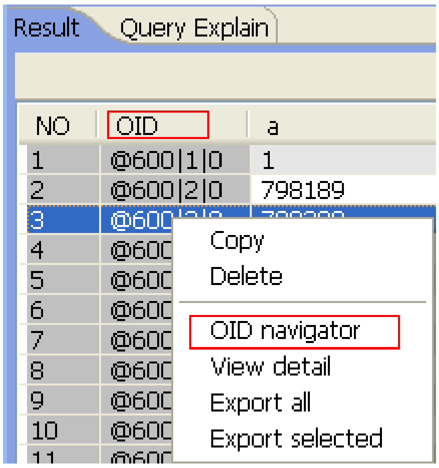
Edit field's value: The information can be modified directly from the query results panel. Change data to a modifiable state by double-clicking it, and then modify it. This option is available when [Get OID info] is set in the given host's Properties ( ) > [Query] dialog box, or [OID info ( )] is specified.



NO	OID	code	name	gender	nation_code	event
1	@680|1|0	10999	Fernandez Jesus	M	ESP	Handball
2	@680|2|0	10998	Fernandez Jaime	M	AUS	Rowing
3	@680|3|0	10997	Fernandez Isabel	W	ESP	Judo
4	@680|4|0	10996	Fernandez Gigi	W	USA	Tennis
5	@680|5|0	10995	Fernandez Ana Ibis	W	CUB	Volleyball
6	@680|6|0	10994	Fernandez Abelardo	M	ESP	Football
7	@680|7|0	10993	Feri Attila	M	HUN	Weightlifting
8	@680|8|0	10992	Felisiak Robert	M	GER	Fencing

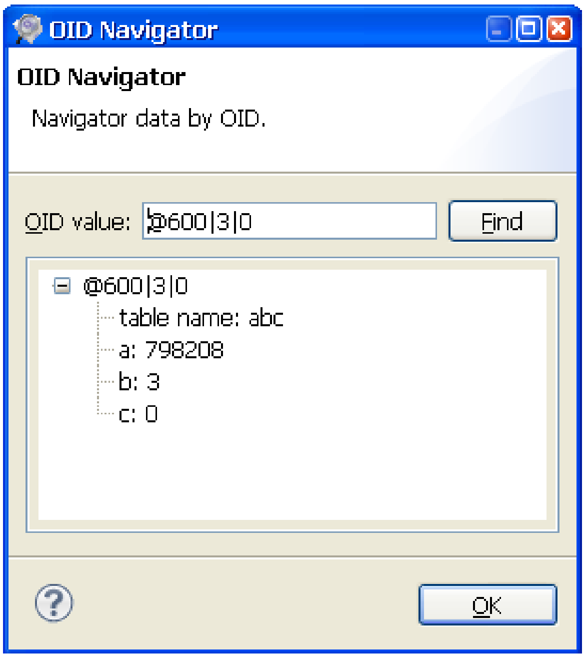
Delete: You can delete data directly in the query results panel. Right-click the row to delete and then select [Delete]. This option is available only when [Get OID info] is set as the modify function does.

OID navigator: This option can be activated by right-clicking an OID data in the [OID] column as follows. It provides a function that allows you to navigate the selected OID directly.



NO	OID	a
1	@600|1|0	1
2	@600|2|0	798189
3	@600	
4	@600	
5	@600	
6	@600	
7	@600	
8	@600	
9	@600	
10	@600	

And the OID navigator as follows will be poped:



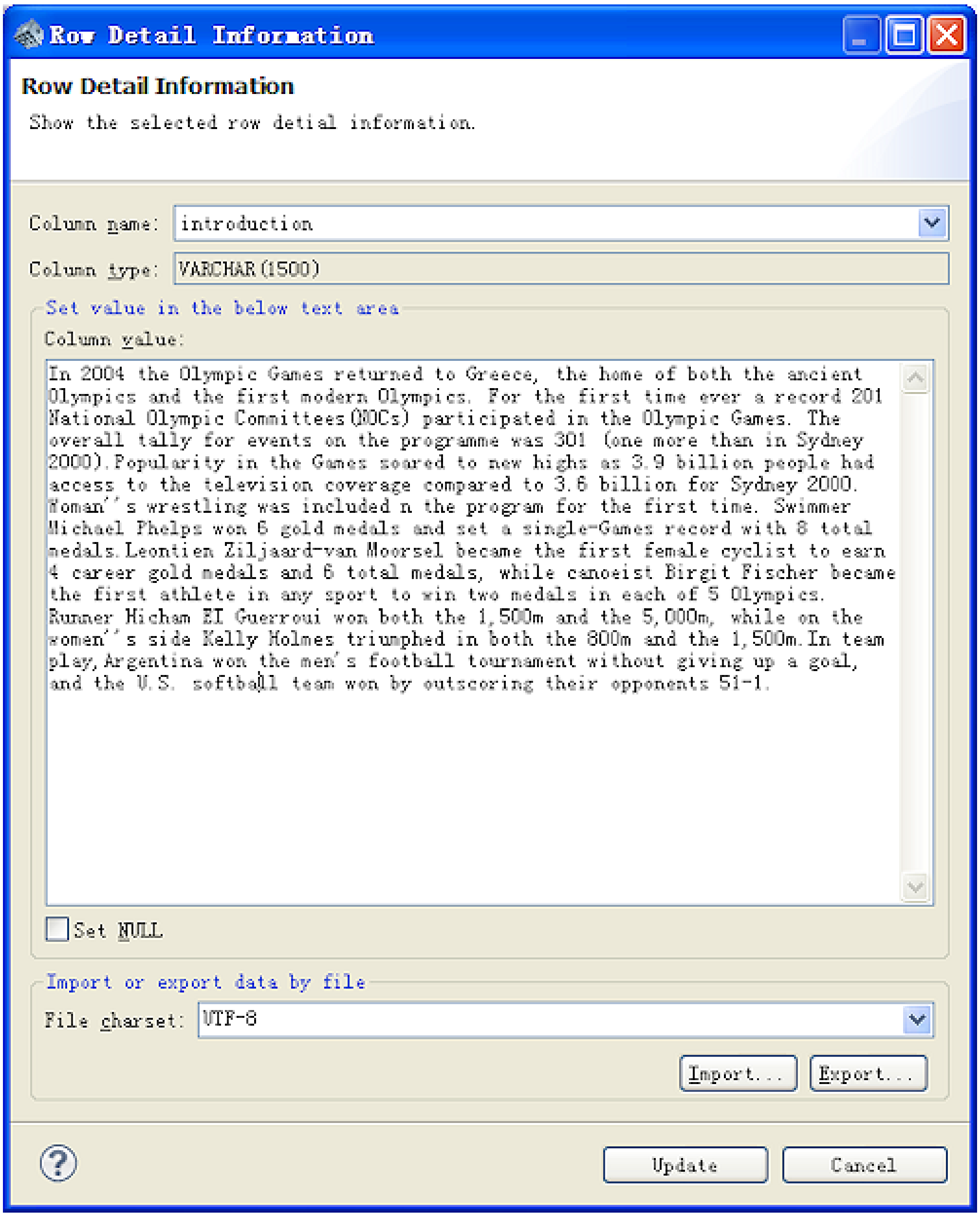
OID Navigator

Navigator data by OID.

OID value: @600|3|0

@600|3|0
- table name: abc
- a: 798208
- b: 3
- c: 0

View detail: If there are too long data to be displayed in a single row, or if its size is too big, it is very difficult to see column values on the query results panel. In this case, you can see column values in detail by right-clicking the row and then selecting [View detail]. You can also modify data directly in the [Detail] dialog box if [Get OID info] option is set. For BLOB or CLOB type data, you can import and export it with a separate file. When column data type is varchar and similar varchar such as char, nchar and etc, file type should be (.txt), if the column data type is another type except BLOB/CLOB, the import/export button will be disabled. And if column data type is BLOB/CLOB, the file type should be the right file type which depends upon the real data type.



Row Detail Information

Show the selected row detial information.

Column name: introduction

Column type: VARCHAR(1500)

Set value in the below text area

Column value:

In 2004 the Olympic Games returned to Greece, the home of both the ancient Olympics and the first modern Olympics. For the first time ever a record 201 National Olympic Committees(NOCs) participated in the Olympic Games. The overall tally for events on the programme was 301 (one more than in Sydney 2000).Popularity in the Games soared to new highs as 3.9 billion people had access to the television coverage compared to 3.6 billion for Sydney 2000. Woman''s wrestling was included n the program for the first time. Swimmer Michael Phelps won 6 gold medals and set a single-Games record with 8 total medals.Leontien Ziljaard-van Moorsel became the first female cyclist to earn 4 career gold medals and 6 total medals, while canoeist Birgit Fischer became the first athlete in any sport to win two medals in each of 5 Olympics. Runner Hicham El Guerroui won both the 1,500m and the 5,000m, while on the women''s side Kelly Holmes triumphed in both the 800m and the 1,500m.In team play,Argentina won the men's football tournament without giving up a goal, and the U.S. softball team won by outscoring their opponents 51-1.

□ Set NULL

Import or export data by file

File charset: UTF-8

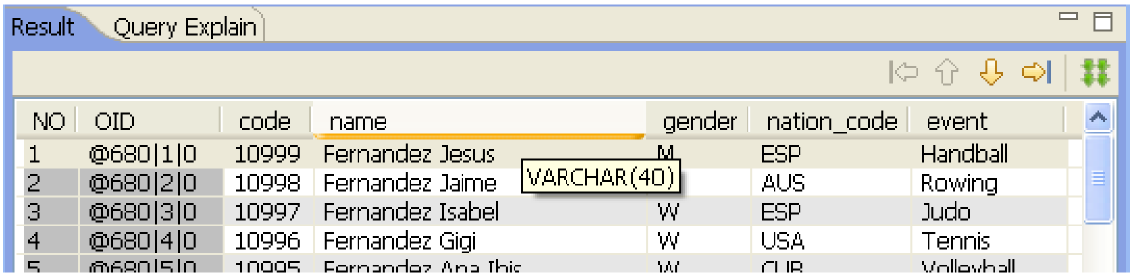
Export all: Exports all data in the query results panel to an Excel or a CSV file.

Export selected: Exports only the data in the row selected in the query results panel to an Excel or a CSV file. The char-set for a file where exported data is saved can be specified.

Export data: Exporting/importing data for **BLOB**, **CLOB**, **BIT VARYING** (>100) cannot be executed, but importing detailed information in the result panel of the Query Editor is possible. Note that only type tags such as (BLOB), (CLOB), and (BIT) are displayed in the result panel, instead of displaying real data.

Import data: If field type is **STRING**, **CHAR**, **VARCHAR**, **NCHAR**, **NCHAR VARYING**, **BIT VARYING**, **BIT, BLOB**, or **CLOB**. The **[Import]** button will be enabled and users can change the field value by specify a file.

When users place the mouse pointer over a column name of the query results panel, a hint about current column's type and size will be popped.



NO	OID	code	name	gender	nation_code	event
1	@680|1|0	10999	Fernandez Jesus	M	ESP	Handball
2	@680|2|0	10998	Fernandez Jaime		AUS	Rowing
3	@680|3|0	10997	Fernandez Isabel	W	ESP	Judo
4	@680|4|0	10996	Fernandez Gigi	W	USA	Tennis
5	@680|5|0	10995	Fernandez Ana Ibis	W	CUB	Volleyball

Query Execution Plan

If [Display query plan] is selected in the Query Editor's options, [Display query plan] becomes activated in the Query Editor Toolbar.

You can check how the selected query will be executed, even without executing it, by clicking [Display query plan] from the toolbar or by clicking < Ctrl+L>. You can also check query execution plans that have already been executed.

Since query plans are displayed all the time at the bottom of the Query Editor window, you can check the existing query plans by opening query plan history files without connecting to the database.

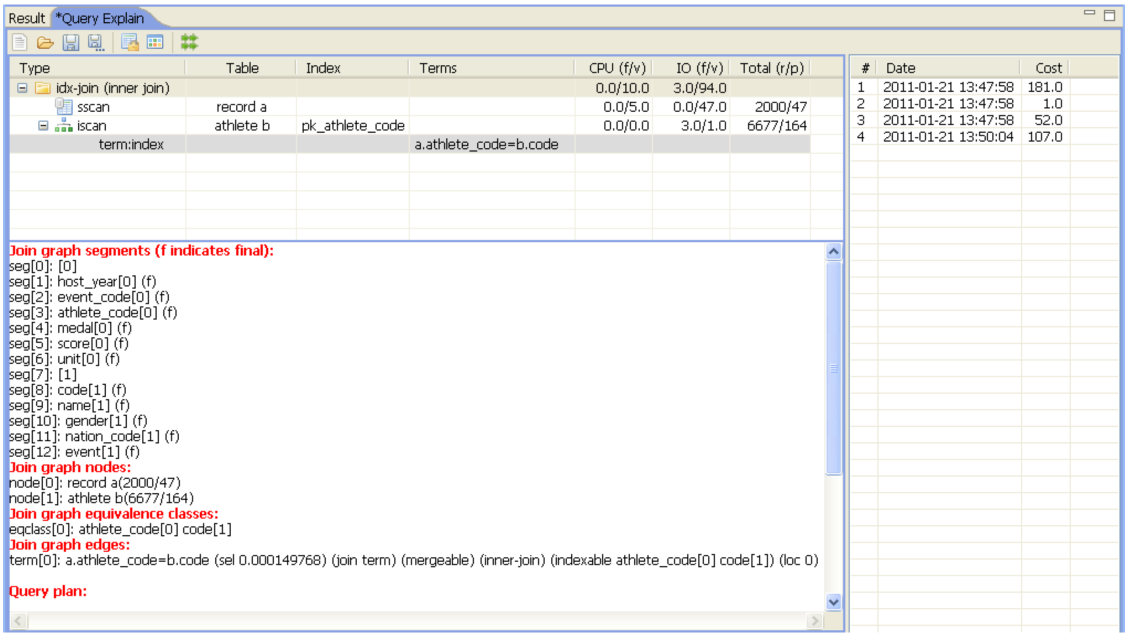
The query plan function retrieves the SQL execution plans it is used not for a one-time purpose but for a collection purpose to continuously manage and retrieve them. Every time you retrieve a query plan, the query plan history is accumulated. You can save this accumulated data to an .xml file. When you open the saved .xml file, you can check the original query plan and executed SQL statements. If it is a one-time retrieval, select [Disable to collecting histories] to view the current query plan without recording history.

The [Query Explain] tab consists of a toolbar, query plan display panel, original statement display panel and query plan history panel.

Query Explain tab

If you select a query and then click [Display query plan], the query plan is displayed in the [Query Explain] tab. The [Query Explain] tab shows the query plan summarized in the tree structure.

The [Query Explain] tab is located to the right of the [Result] tab. You can switch to the [Query Explain] tab while viewing the query result.



Type	Table	Index	Terms	CPU (f/v)	IO (f/v)	Total (r/p)
idx-join (inner join)				0.0/10.0	3.0/94.0	
sscan	record a			0.0/5.0	0.0/47.0	2000/47
iscan	athlete b	pk_athlete_code		0.0/0.0	3.0/1.0	6677/164
term:index			a.athlete_code=b.code			

#	Date	Cost
1	2011-01-21 13:47:58	181.0
2	2011-01-21 13:47:58	1.0
3	2011-01-21 13:47:58	52.0
4	2011-01-21 13:50:04	107.0

```
Join graph segments (f indicates final):
seg[0]: [0]
seg[1]: host_year[0] (f)
seg[2]: event_code[0] (f)
seg[3]: athlete_code[0] (f)
seg[4]: medal[0] (f)
seg[5]: score[0] (f)
seg[6]: unit[0] (f)
seg[7]: [1]
seg[8]: code[1] (f)
seg[9]: name[1] (f)
seg[10]: gender[1] (f)
seg[11]: nation_code[1] (f)
seg[12]: event[1] (f)
Join graph nodes:
node[0]: record a(2000/47)
node[1]: athlete b(6677/164)
Join graph equivalence classes:
eqclass[0]: athlete_code[0] code[1]
Join graph edges:
term[0]: a.athlete_code=b.code (sel 0.000149768) (join term) (mergeable) (inner-join) (indexable athlete_code[0] code[1]) (loc 0)

Query plan:
```

Query Explain Toolbar

The Query Explain toolbar has the following functions:

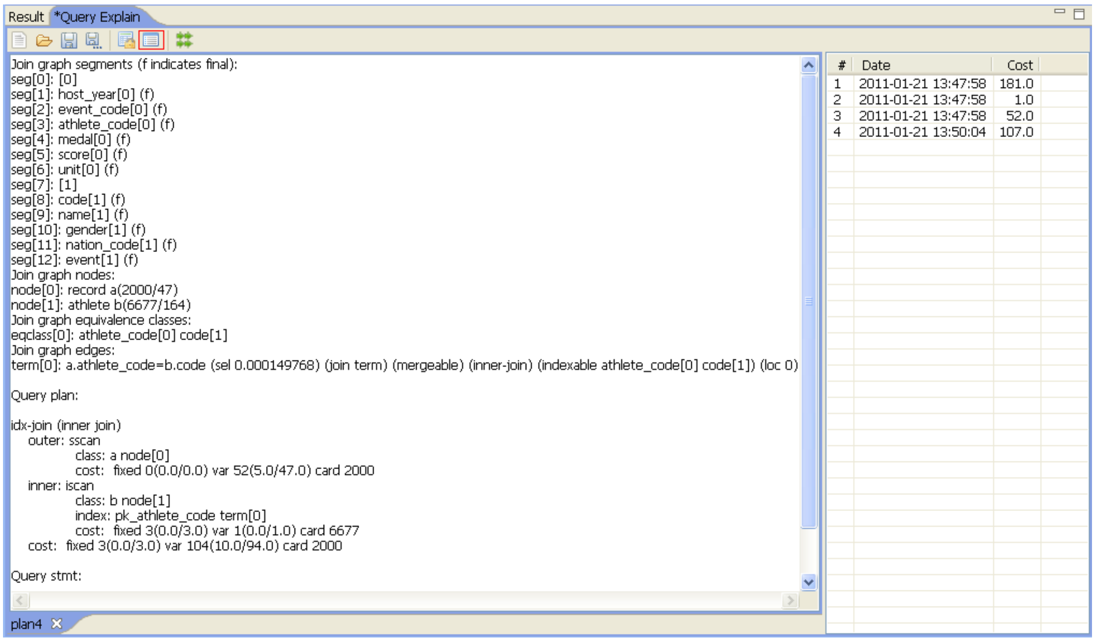
New : Initializes all query plan histories retrieved so far and begins collecting them again. This function is used to initialize the existing job.

Open a query explain file : Imports a previously saved query plan history file. If you open the .xml file, you can view the original query plan.

Save a query explain file . **Save a query explain file as** : Saves the collected query plan to an external file. The file extension is **xml**.

Disable to collecting histories : Histories are added to the query plan history panel whenever you retrieve the execution plan by using [Display query plan] in the Query Editor. To retrieve a query plan temporarily without recording history, click [Disable to collecting histories]. The name of the tab below the query plan display panel is displayed as "plan."

Query plan type : By toggling the icon, the type can be switched between text and tree . You can view the unprocessed query plan source created by the database in a text form.

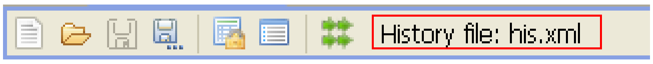


```
Join graph segments (f indicates final):
seg[0]: [0]
seg[1]: host_year[0] (f)
seg[2]: event_code[0] (f)
seg[3]: athlete_code[0] (f)
seg[4]: medal[0] (f)
seg[5]: score[0] (f)
seg[6]: unit[0] (f)
seg[7]: [1]
seg[8]: code[1] (f)
seg[9]: name[1] (f)
seg[10]: gender[1] (f)
seg[11]: nation_code[1] (f)
seg[12]: event[1] (f)
Join graph nodes:
node[0]: record a(2000/47)
node[1]: athlete b(6677/164)
Join graph equivalence classes:
eqclass[0]: athlete_code[0] code[1]
Join graph edges:
term[0]: a.athlete_code=b.code (sel 0.000149768) (join term) (mergeable) (inner-join) (indexable athlete_code[0] code[1]) (loc 0)

Query plan:

idx-join (inner join)
    outer: sscan
            class: a node[0]
            cost:  fixed 0(0.0/0.0) var 52(5.0/47.0) card 2000
    inner: iscan
            class: b node[1]
            index: pk_athlete_code term[0]
            cost:  fixed 3(0.0/3.0) var 1(0.0/1.0) card 6677
    cost:  fixed 3(0.0/3.0) var 104(10.0/94.0) card 2000

Query stmt:
```

#	Date	Cost
1	2011-01-21 13:47:58	181.0
2	2011-01-21 13:47:58	1.0
3	2011-01-21 13:47:58	52.0
4	2011-01-21 13:50:04	107.0

Show or Hide a query history panel ◧ ▷: Shows or hides the collection history panel on the right side.

Query plan history file: If a query plan history is saved as a file, or an existing file is opened, the name of the currently used query plan history file is displayed.

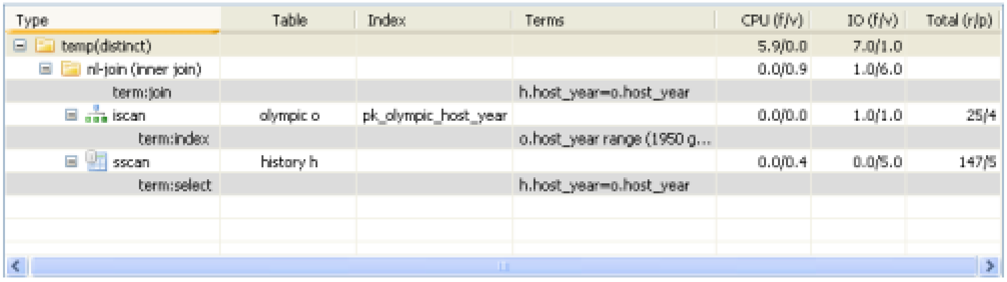


Query Plan Display

In the query plan display panel, the query plan executed in each step is displayed in the tree structure.

Each item in the vertical axis is called a node. Each node contains different data. You can view the tree moving from the top to the bottom.

The horizontal axis is called an item and contains Type, Table, Index, Terms, CPU I/O cost, Disk I/O cost, and Total (ROW/PAGE).



Type	Table	Index	Terms	CPU (f/v)	IO (f/v)	Total (r/p)
⊟ temp(distinct)				5.9/0.0	7.0/1.0	
⊟ nl-join (inner join)				0.0/0.9	1.0/6.0	
term:join			h.host_year=o.host_year			
⊟ iscan	olympic o	pk_olympic_host_year		0.0/0.0	1.0/1.0	25/4
term:index			o.host_year range (1950 g...			
⊟ sscan	history h			0.0/0.4	0.0/5.0	147/5
term:select			h.host_year=o.host_year			

Type: Indicates the scan type or join method (such as **sscan**, **iscan** and **idx-join**).

Table: The table (class), view (virtual class) and alias, which are referred to when the node is executed, are displayed altogether.

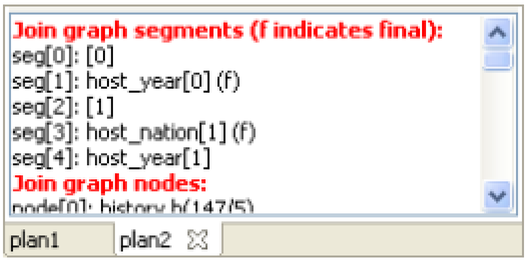
Index: The name of the index used is displayed if the type is **iscan**.

Terms: Join and filter conditions are displayed. The contents are hidden for readability. If you click [+] button, sub-nodes will be extended to show the details. In addition, different colors are used for different search conditions.

Cost: Displays CPU and Disk I/O cost of the query plan. Fixed and variable costs are displayed separately.

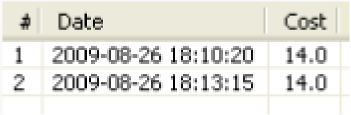
Total (r/p): Displays the total number of rows and the number of pages to be used to fetch data.

The original statements of the query plan selected in the query plan panel are displayed below the query plan display panel.



```
Join graph segments (f indicates final):
seg[0]: [0]
seg[1]: host_year[0] (f)
seg[2]: [1]
seg[3]: host_nation[1] (f)
seg[4]: host_year[1]
Join graph nodes:
node[0]: history h(147/5)
```

Query Plan History: Histories are displayed accumulatively in the query plan history panel every time a query plan is executed. # is the accumulation order and corresponds to the tab number below the query plan display panel. Date indicates the date when a query plan is executed, and Cost is the sum of CPU and Disk I/O costs. If you double-click an item in the query plan history panel, you can view the query plan again in the query plan display panel.

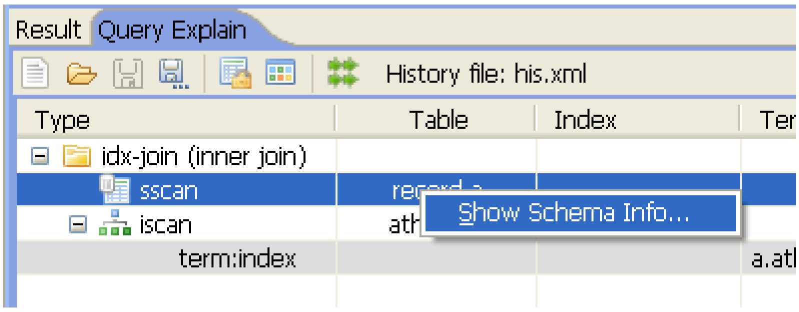


#	Date	Cost
1	2009-08-26 18:10:20	14.0
2	2009-08-26 18:13:15	14.0

Using Query Plans

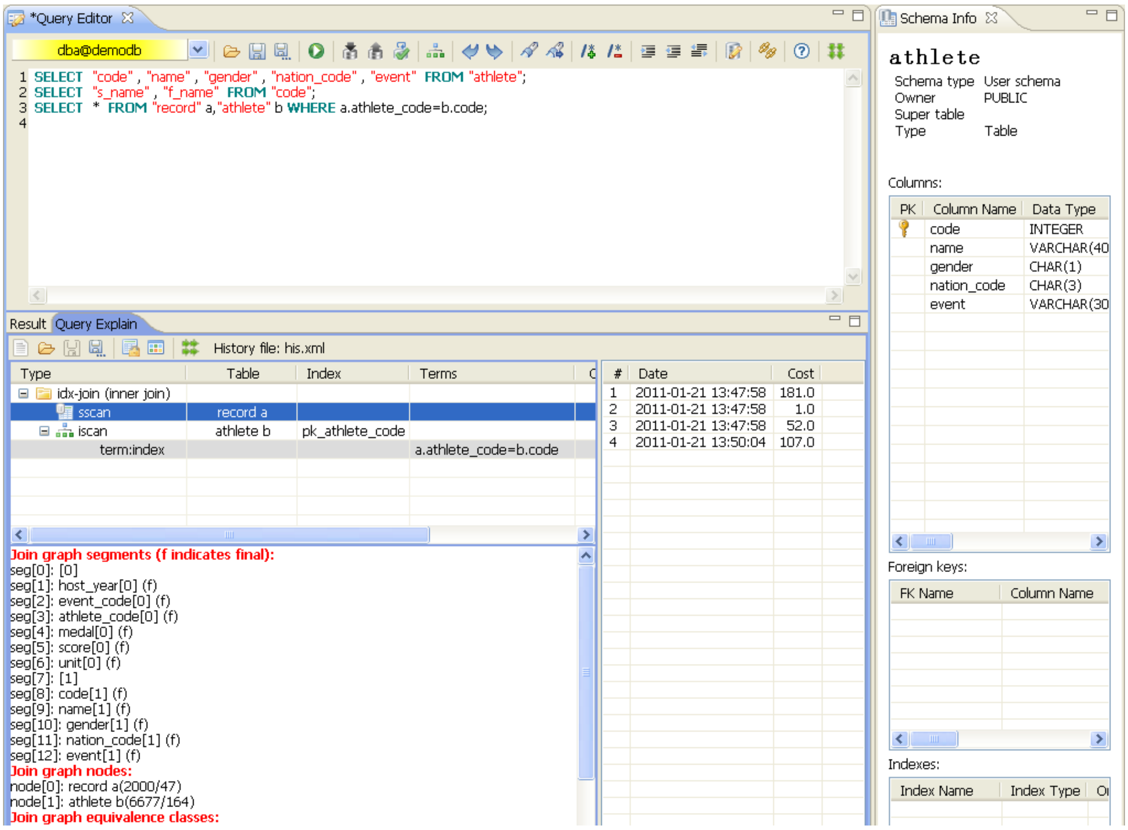
By using the query plan function, you can analyze data while viewing the query plan and the schema info of the corresponding table.

If you right-click a row where a table is located in the query plan display panel and then select [Show Schema Info], you can open and view the schema information of the table as well.



Type	Table	Index	Ter
idx-join (inner join)			
sscan	record a		
iscan	ath		
term:index			a.atl

You can also view the information in a separate window by dragging the Schema Info panel out of the CUBRID Manager. This can be useful in an environment using multiple monitors.



```
SELECT "code" , "name" , "gender" , "nation_code" , "event" FROM "athlete";
SELECT "s_name" , "f_name" FROM "code";
SELECT * FROM "record" a,"athlete" b WHERE a.athlete_code=b.code;
```

Type	Table	Index	Terms
idx-join (inner join)			
sscan	record a		
iscan	athlete b	pk_athlete_code	
term:index			a.athlete_code=b.code

#	Date	Cost
1	2011-01-21 13:47:58	181.0
2	2011-01-21 13:47:58	1.0
3	2011-01-21 13:47:58	52.0
4	2011-01-21 13:50:04	107.0

```
Join graph segments (f indicates final):
seg[0]: [0]
seg[1]: host_year[0] (f)
seg[2]: event_code[0] (f)
seg[3]: athlete_code[0] (f)
seg[4]: medal[0] (f)
seg[5]: score[0] (f)
seg[6]: unit[0] (f)
seg[7]: [1]
seg[8]: code[1] (f)
seg[9]: name[1] (f)
seg[10]: gender[1] (f)
seg[11]: nation_code[1] (f)
seg[12]: event[1] (f)
Join graph nodes:
node[0]: record a(2000/47)
node[1]: athlete b(6677/164)
Join graph equivalence classes:
```

athlete

Schema type	User schema
Owner	PUBLIC
Super table	
Type	Table

Columns:

PK	Column Name	Data Type
🔑	code	INTEGER
	name	VARCHAR(40
	gender	CHAR(1)
	nation_code	CHAR(3)
	event	VARCHAR(30

Foreign keys:

FK Name	Column Name

Indexes:

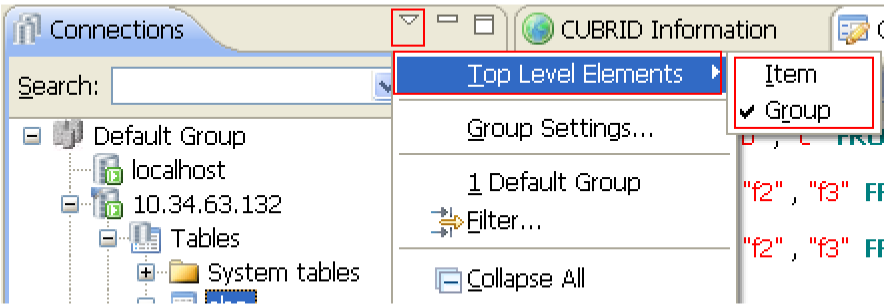
Index Name	Index Type	O

Tasks

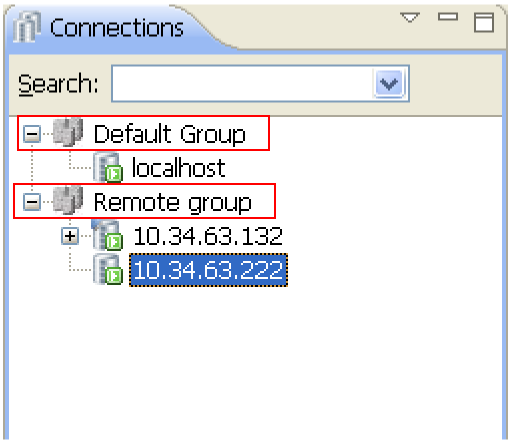
Connection Group

Change display style

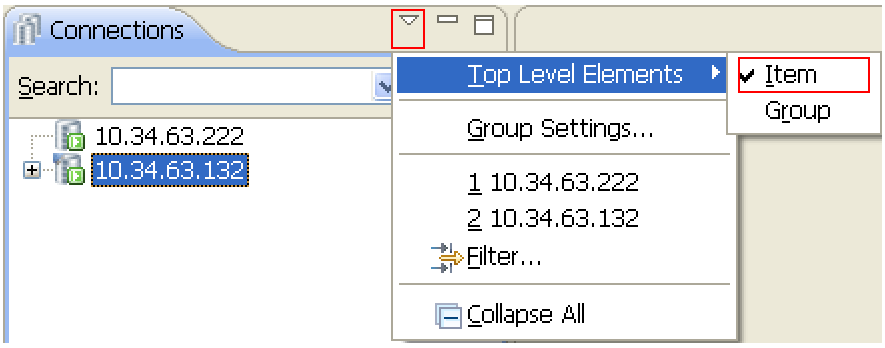
If the [Group] menu is checked as follows:



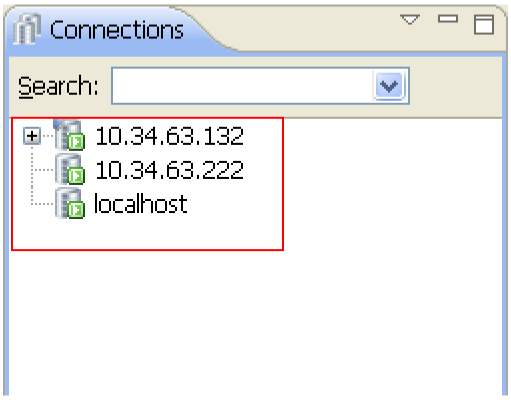
The connection explorer will be as follows:



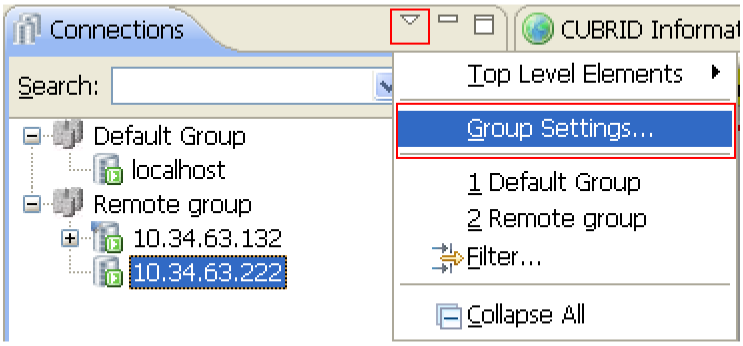
If the [Item] menu is checked as follows:



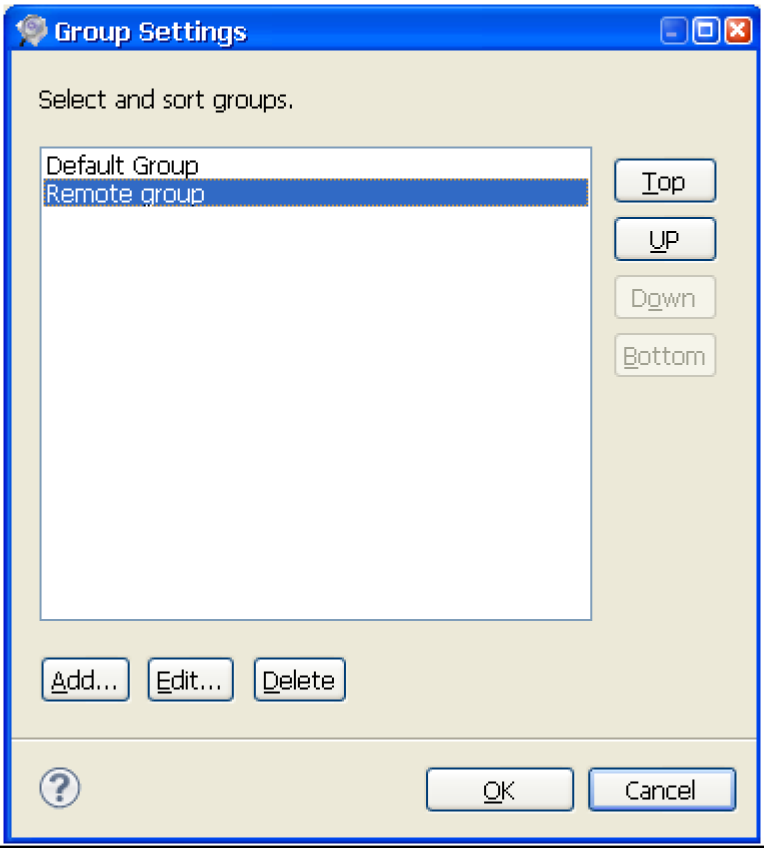
The connection explorer will be as follows:



Add Group



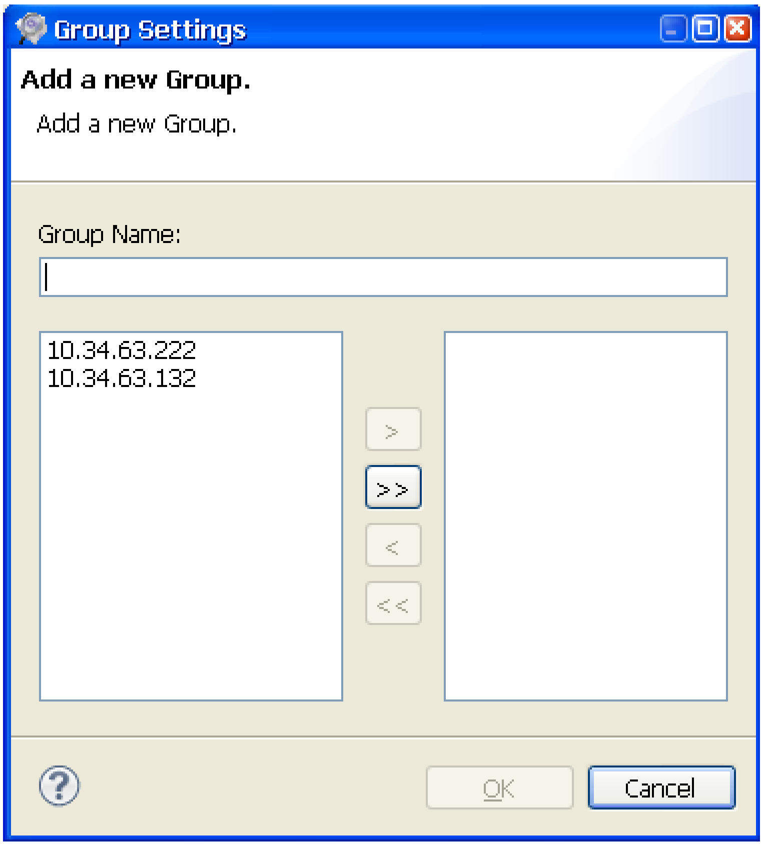
The **[Group Settings]** menu will open a group management dialog:


Group Settings

Select and sort groups.

The default group cannot be edited and deleted. It stores all connections that are not in other groups.

Click the **[Add]** button, a adding group dialog will be popped:



Group Settings

Add a new Group.

Add a new Group.

Group Name:

10.34.63.222
10.34.63.132

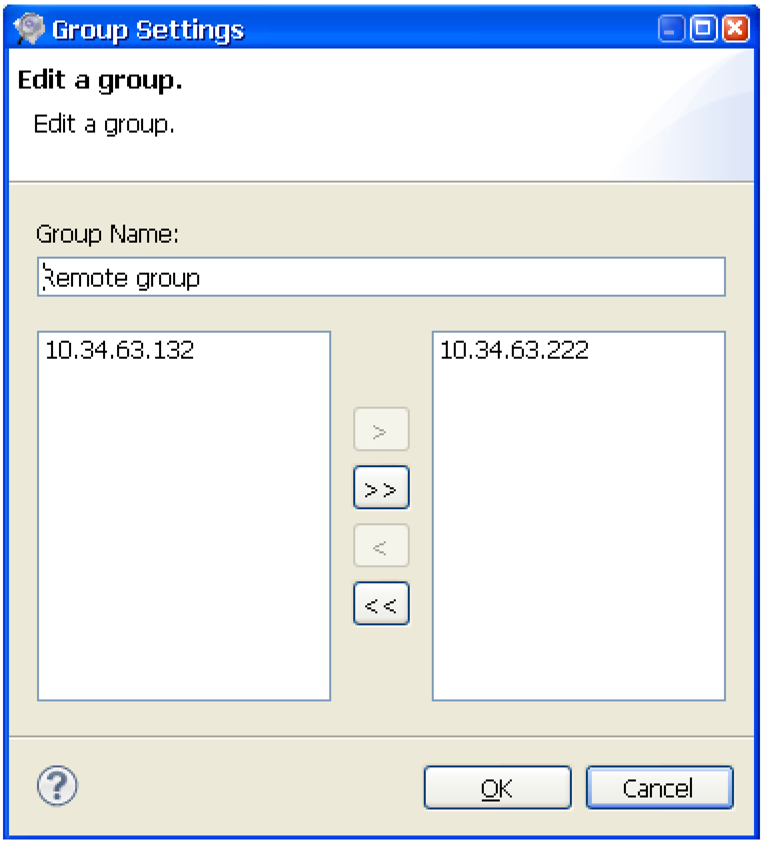
The group name is required and the name cannot be duplicated.

Moving the connections from left to right can make the connections belong to current new group.

Moving connections from right to left will remove the current group’s children connections.

Edit Group

Open the [Dialog Setting] dialog and select a group and click button [Edit]:



The name cannot be duplicated.

The default group cannot be edited.

Delete Group

Open the [Dialog Setting] dialog and select a group and click button [Delete].

The children connections of group to be deleted will be the children of default group.

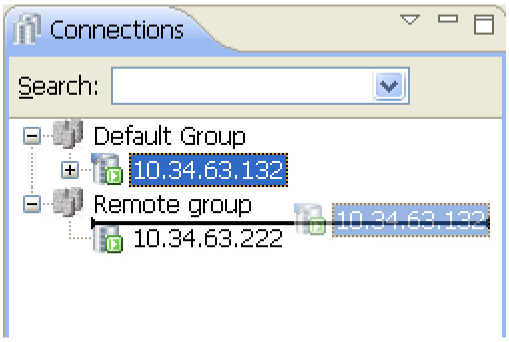
The default group cannot be deleted.

Change display order of groups

Open the [Dialog Setting] dialog and select a group and change its display order by buttons [Top], [Up], [Down], [Bottom].

Drag and Drop

The connection explorer supports that a connection or connections are dragged from one group and dropped to another group, after the dropping, connections will be children of the new group.



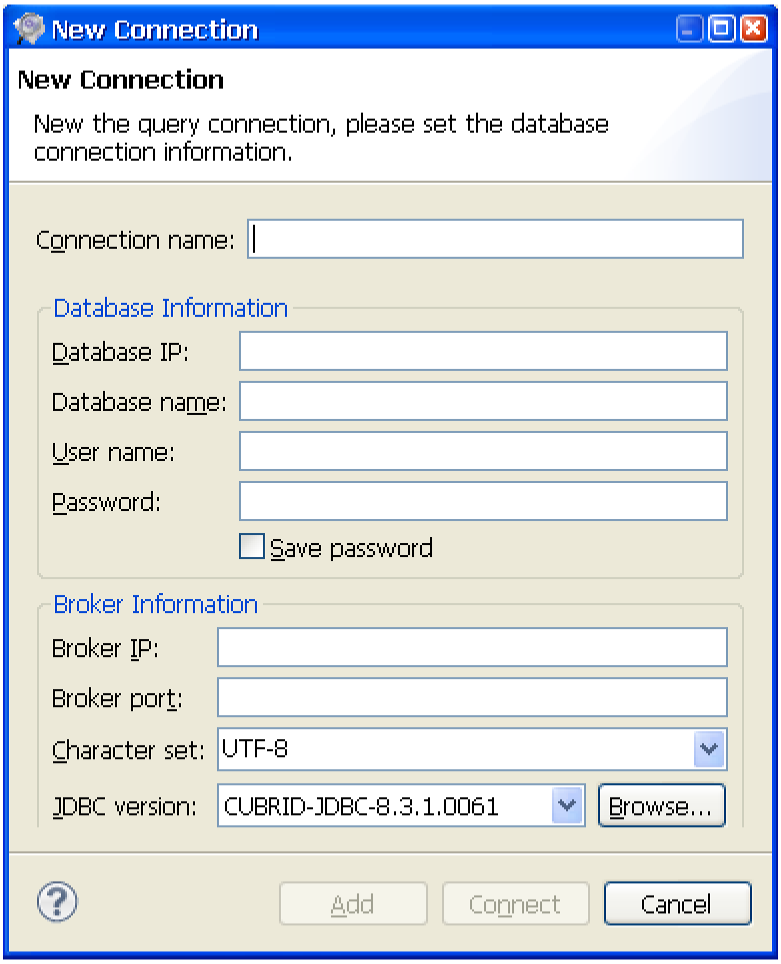
Database Connection

Add Connection

There are lots of ways to add a new connection.

It can be selecting the **[File]→[New Connection]** menu or **[New Connection]** tool button or **[New Connection]** menu in context menu of group and connection nodes.

The new connection dialog will be popped when the menu selected:


New Connection

New Connection

New the query connection, please set the database connection information.

Connection name:

Database Information

Database IP:

Database name:

User name:

Password:

□ Save password

Broker Information

Broker IP:

Broker port:

Character set: UTF-8

JDBC version: CUBRID-JDBC-8.3.1.0061

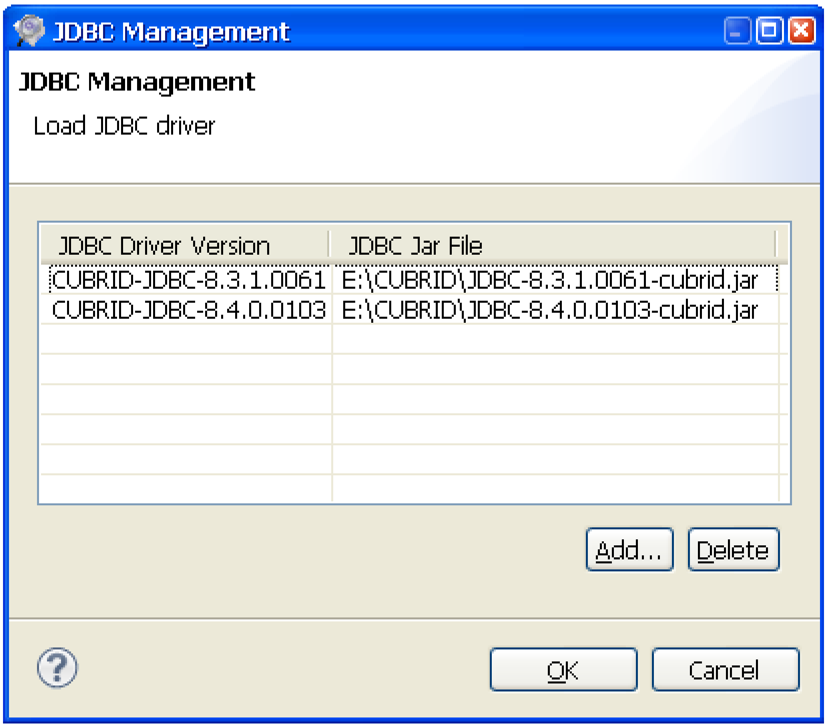
All input area cannot be empty except password.

Connection name cannot be duplicated.

If **[Save password]** is checked, next time when double-clicking the connection, the saved password will be used to connect the database.

The JDBC version must be corresponded with the database server's version. For example if the server is 8.3.1 and the 8.3.1.XXXX JDBC version must be choose.

Clicking the **[Browse]**, a dialog will be popped to manage All JDBC drivers.

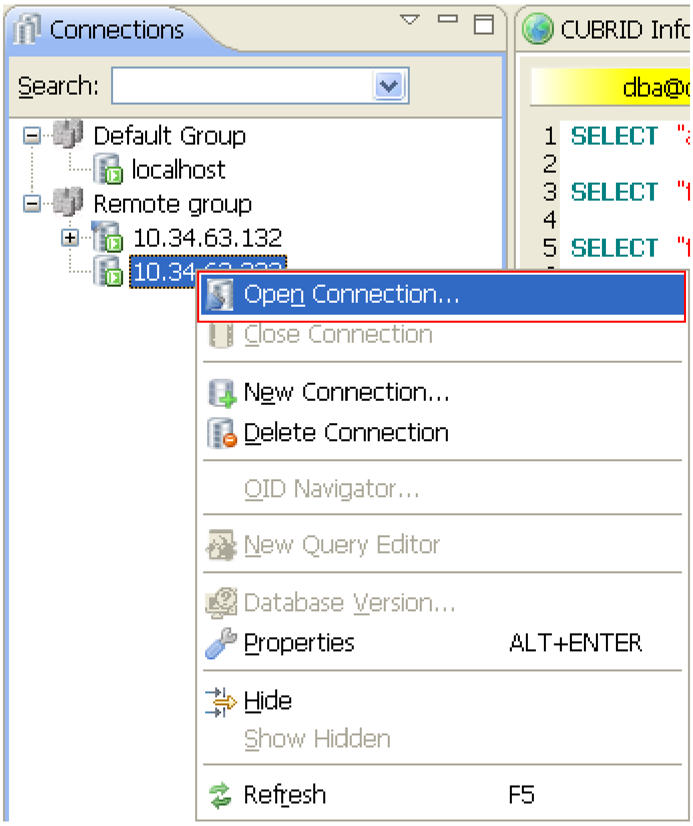


JDBC Driver Version	JDBC Jar File
CUBRID-JDBC-8.3.1.0061	E:\CUBRID\JDBC-8.3.1.0061-cubrid.jar
CUBRID-JDBC-8.4.0.0103	E:\CUBRID\JDBC-8.4.0.0103-cubrid.jar

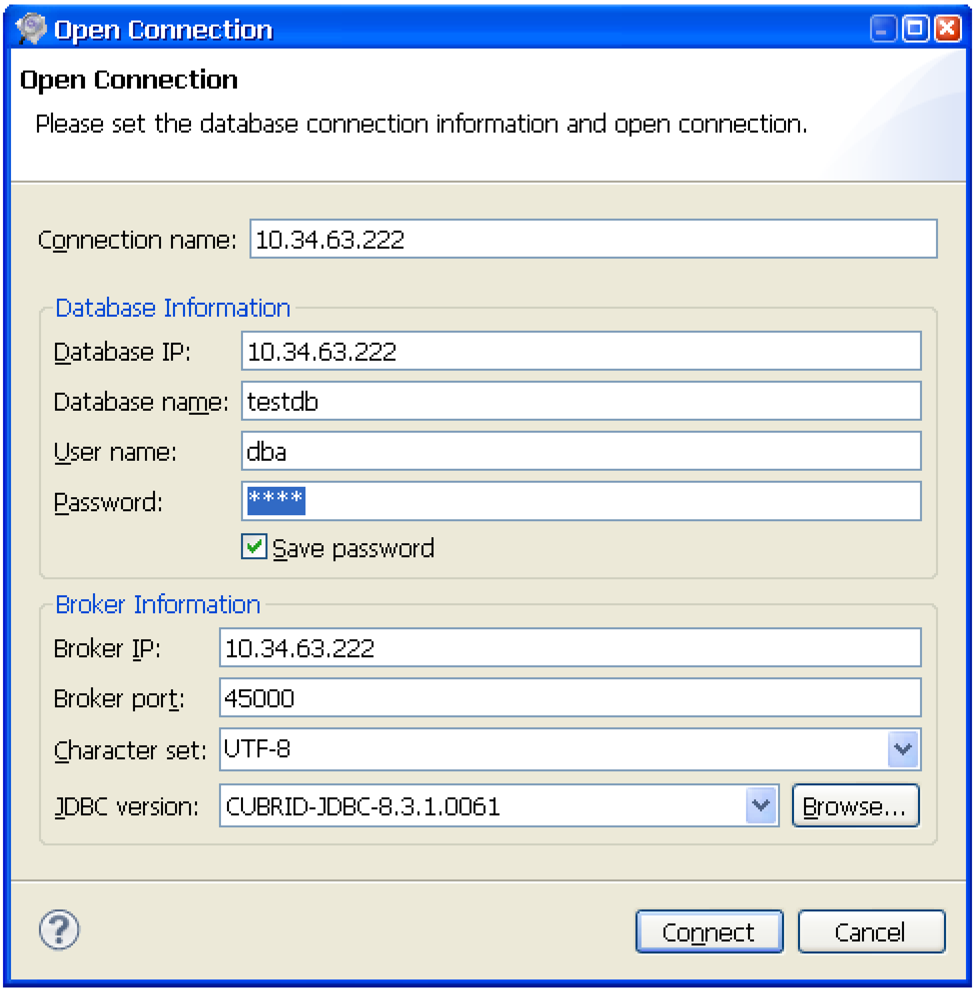
Visit the web site http://www.cubrid.org/cubrid_java_programming to download the CUBRID JDBC drivers.

Open Connection

To open a dialog named “Open Connection” which is used to input connection information, please right-click the connection node and selected the menu [Open Connection]:



And the dialog will be opened as follows:



Open Connection

Open Connection

Please set the database connection information and open connection.

Connection name: 10.34.63.222

Database Information

Database IP: 10.34.63.222

Database name: testdb

User name: dba

Password: ****

☑ Save password

Broker Information

Broker IP: 10.34.63.222

Broker port: 45000

Character set: UTF-8

JDBC version: CUBRID-JDBC-8.3.1.0061

If the **[Save password]** is checked, the password will be saved at local after **[Connect]** button pressed, next, the operation of double click connection node will open the database connection directly without open the dialog.

Delete Connection

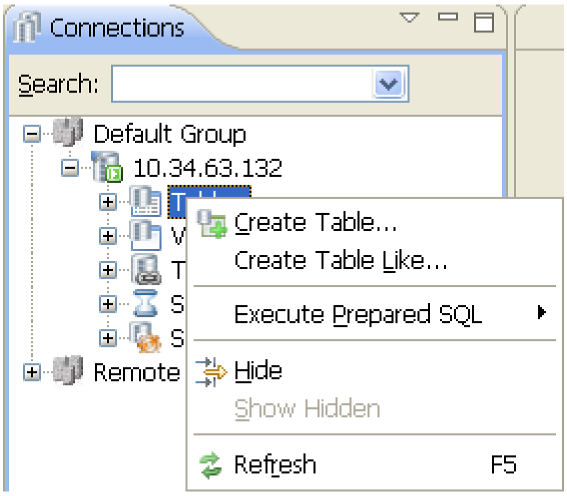
Select a connection and select the **[File]→[Delete Connection]** menu or **[Delete Connection]** menu in the context menu of current selected connection to delete a connection.

Drag and Drop

Connections can be dragged and dropped to change their group or their display order.

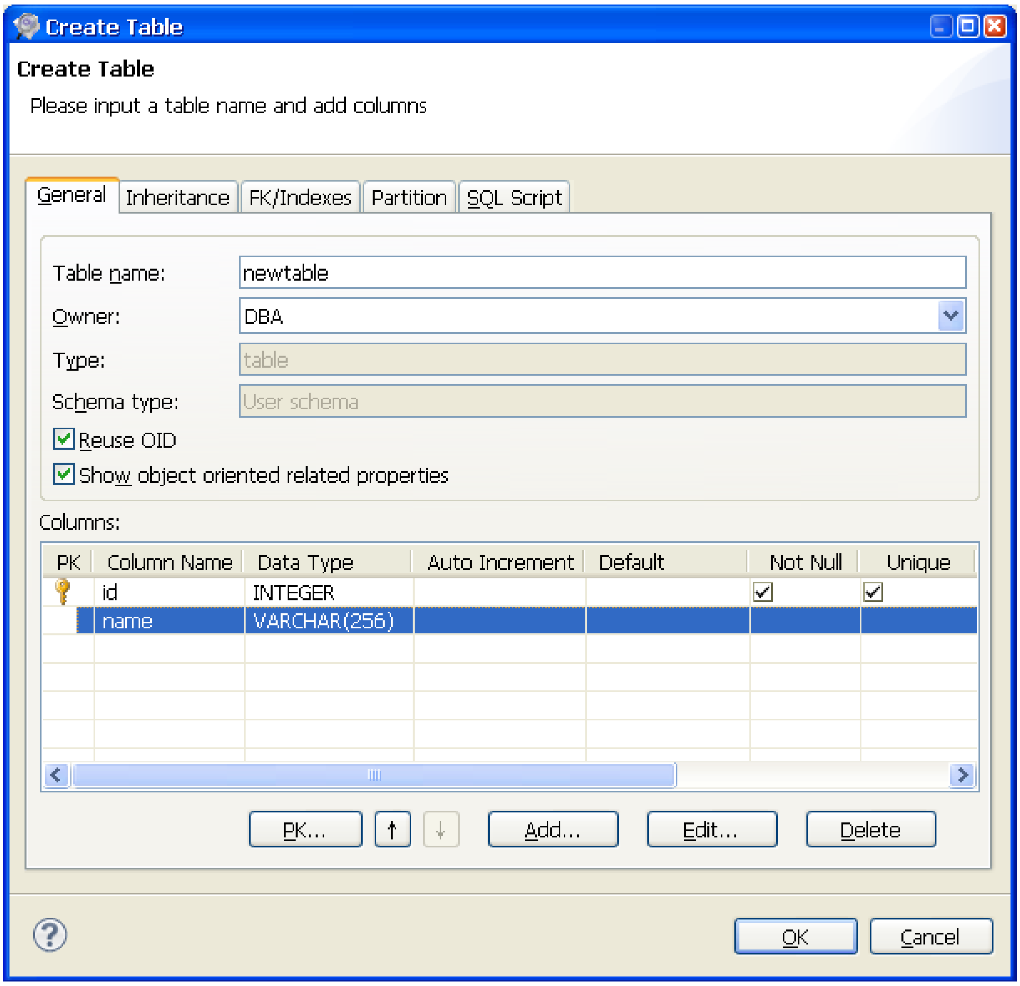
Table

When the database is connected, tables and system tables which current login user has been permitted to visit will be listed under [Table] folder node in the navigation tree. The context menu of the [Table] folder node in the navigation tree is as follows:



Create Table

Right-click [Tables] in the navigation tree and then select [Create Table] or click [Create Table] button in the toolbar. Then, a wizard who makes you create a new table will be popped as follows:



Create Table

Please input a table name and add columns

Table name: newtable

Owner: DBA

Type: table

Schema type: User schema

☑ Reuse OID

☑ Show object oriented related properties

Columns:

PK	Column Name	Data Type	Auto Increment	Default	Not Null	Unique
🔑	id	INTEGER			☑	☑
	name	VARCHAR(256)				

[General] tab: You can define the name of the new table and add, edit and drop columns of it. You also can set the primary key (PK) and, before the table is created, change the order of the selected column by using buttons **[↑]** or **[↓]**.

Using the **[add]** column wizard, you can set the name, type, default value and constraints of a new column. You also can set to display a warning message when you enter something that is not grammatically correct, or to disable the selection of a grammatically incorrect entry.

[Reuse OID]: If this option is selected, a table is created with the option [**Reuse OID]** applied. For this kind of table, using with OID is restricted. For more information on OID reuse table, see Table Options (Reuse OID).

[FK/Indexes] tab: You can set up foreign keys and indexes of the table.

[Add FK]: Using the [add foreign] key wizard, you can set the name of the foreign key, the name and primary key of the reference table and the trigger actions to maintain referential integrity. The **ON UPDATE**, **ON DELETE** and **ON CACHE OBJECT** options are provided.

[Add Index]: You can set index name, index type, a column to be indexed. Ascending (asc) and descending (desc) sorting can be selected within the supported range. For **[REVERSE]** indexes, only descending sorting is supported.

[Partition] Tab: Supports partitioning setting and modification of the given table.

[Add] Partition: Using the add partition wizard, you can set the partition type and expression. The **RANGE**, **LIST** and **HASH** partitions are supported.

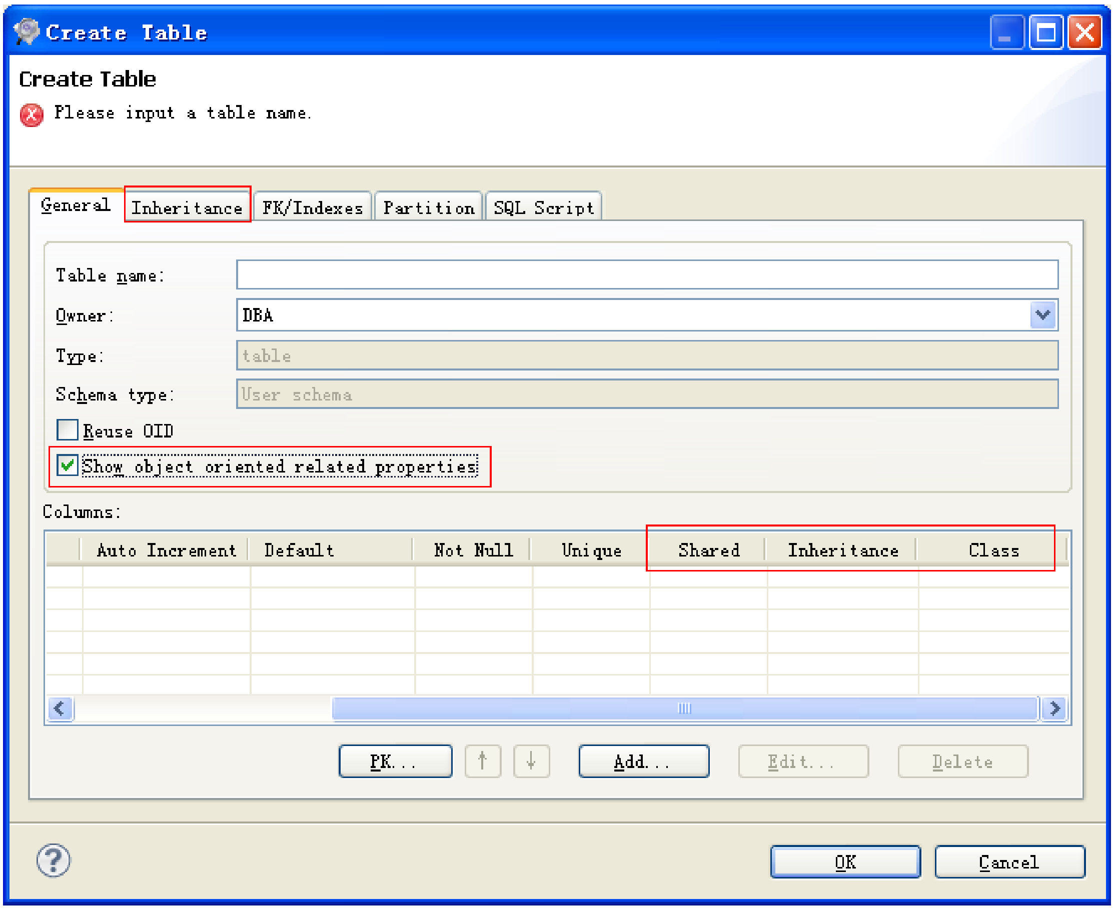
[Edit] Partition: Edit partition.

[Delete] Partition: Delete partition.

[SQL Script] Tab: You can check and copy SQL statements created according to the settings in **[General]**, **[FK/Indexes]** and **[Partition]** tabs.

[Adding Object Oriented Tables]

To add a table with object-oriented properties, select **[Show object oriented related properties]**. When it is checked, the **[Inheritance]** tab is added to tabs.



Create Table

Please input a table name.

Table name:

Owner: DBA

Type: table

Schema type: User schema

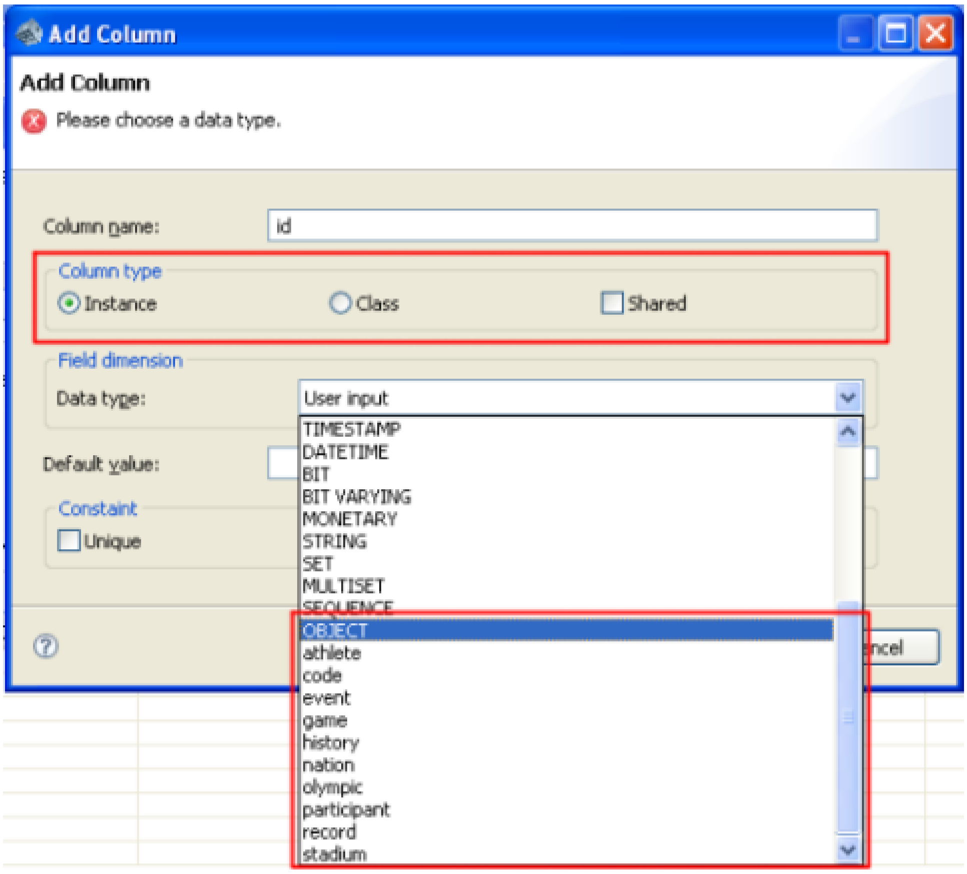
□ Reuse OID

☑ Show object oriented related properties

Columns:

Auto Increment	Default	Not Null	Unique	Shared	Inheritance	Class

[General] tab: When you select [Show object oriented related properties], Shared, Inheritance and Table column are added as a column in the table. When adding or editing columns, you can choose the column type as shown in the figure below. You can choose OBJECT or a table in the database as the data type.



Add Column

Please choose a data type.

Column name: id

Column type

● Instance ○ Class □ Shared

Field dimension

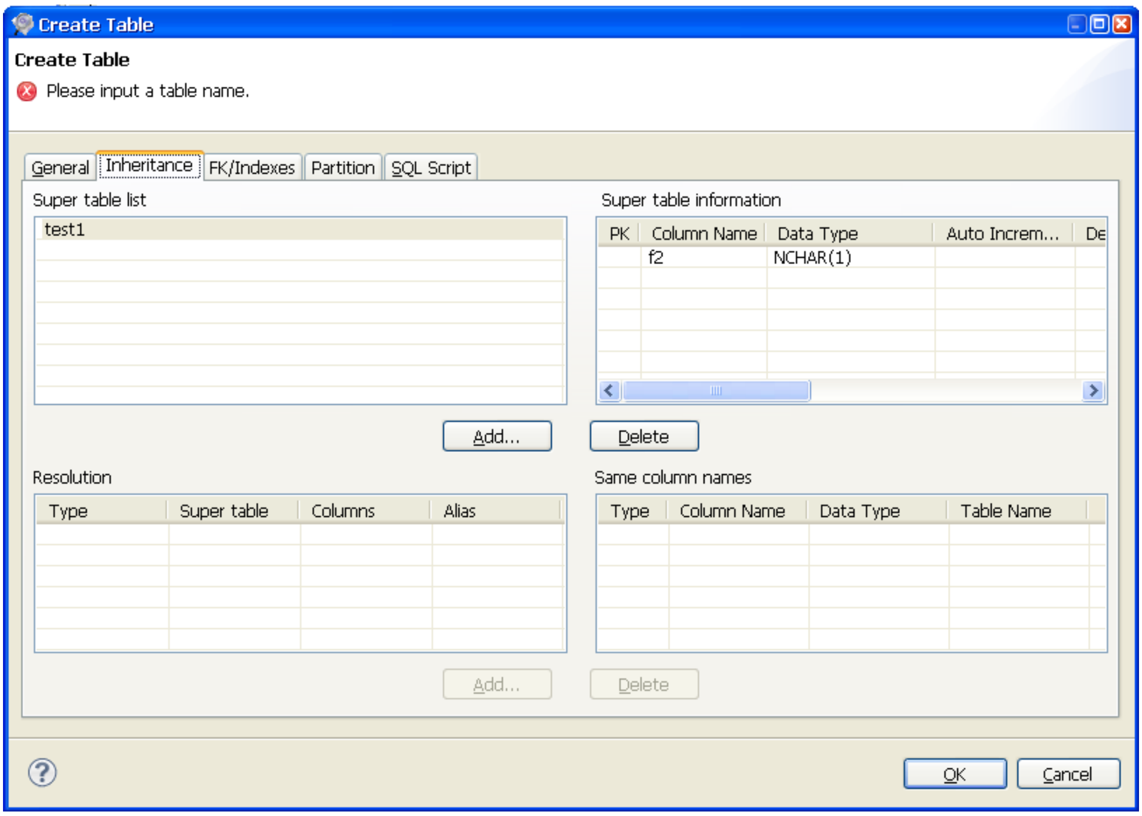
Data type: User input

Default value:

Constraint

□ Unique

[Inheritance] tab: You can define the super table to inherit from. If a column name conflict occurs, it can be adjusted.



Create Table

Create Table

Please input a table name.

Super table list

test1

Super table information

PK	Column Name	Data Type	Auto Increm...	De
	f2	NCHAR(1)		

Resolution

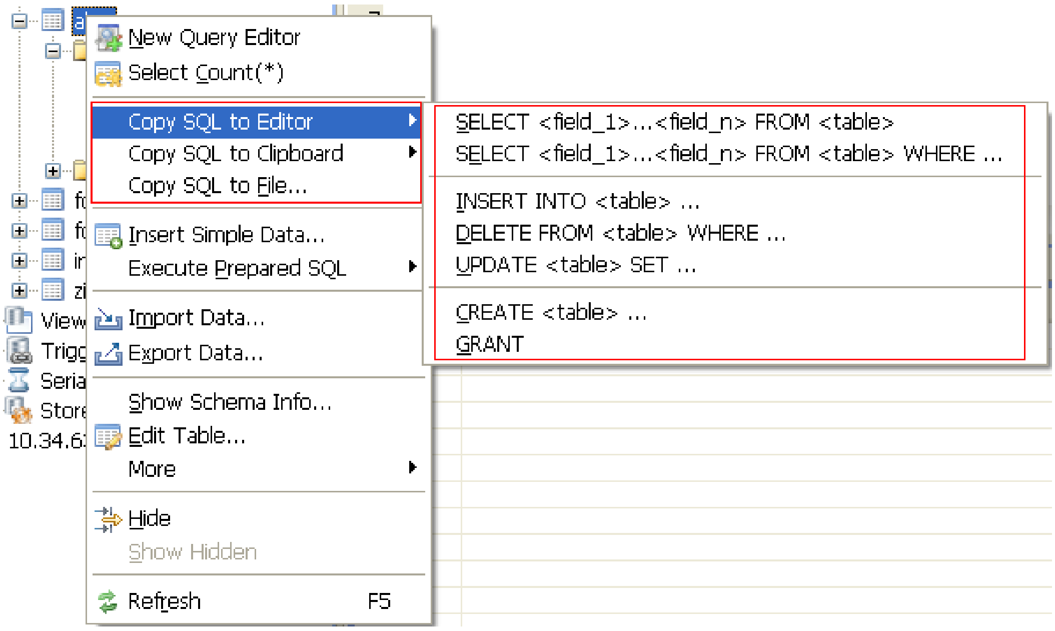
Type	Super table	Columns	Alias

Same column names

Type	Column Name	Data Type	Table Name

Copy SQL

Users can generate SELECT/INSERT/UPDATE/DELETE/CREATE/GRANT SQLs about current selected table(s) and copy them to current query editor, clipboard or a specific file. The menus as follows:



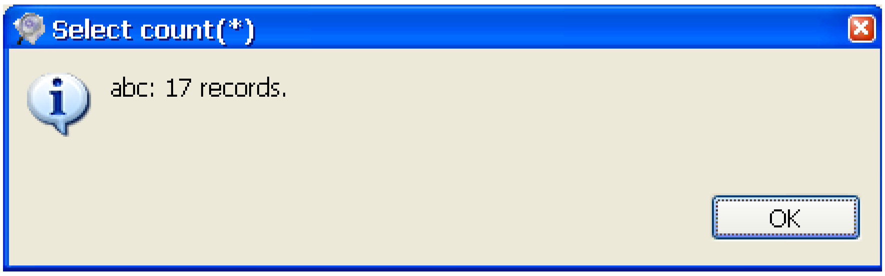
Select All

Right-click a table in the navigation tree and select **[More]** → **[Select all]**. Or you can drag and drop the table into the editor results panel of the Query editor when it is open. Then, a new Query editor opens and retrieves the entire data.

Select Count

Retrieves total row count of the table and performs the same functionality as the following syntax.

SELECT COUNT (*) **FROM** [table-name]



abc: 17 records.

Delete All Records

[More]→[Delete All] menu will delete all records from the table and it is as same as the following SQL statement:

DELETE FROM [table-name]

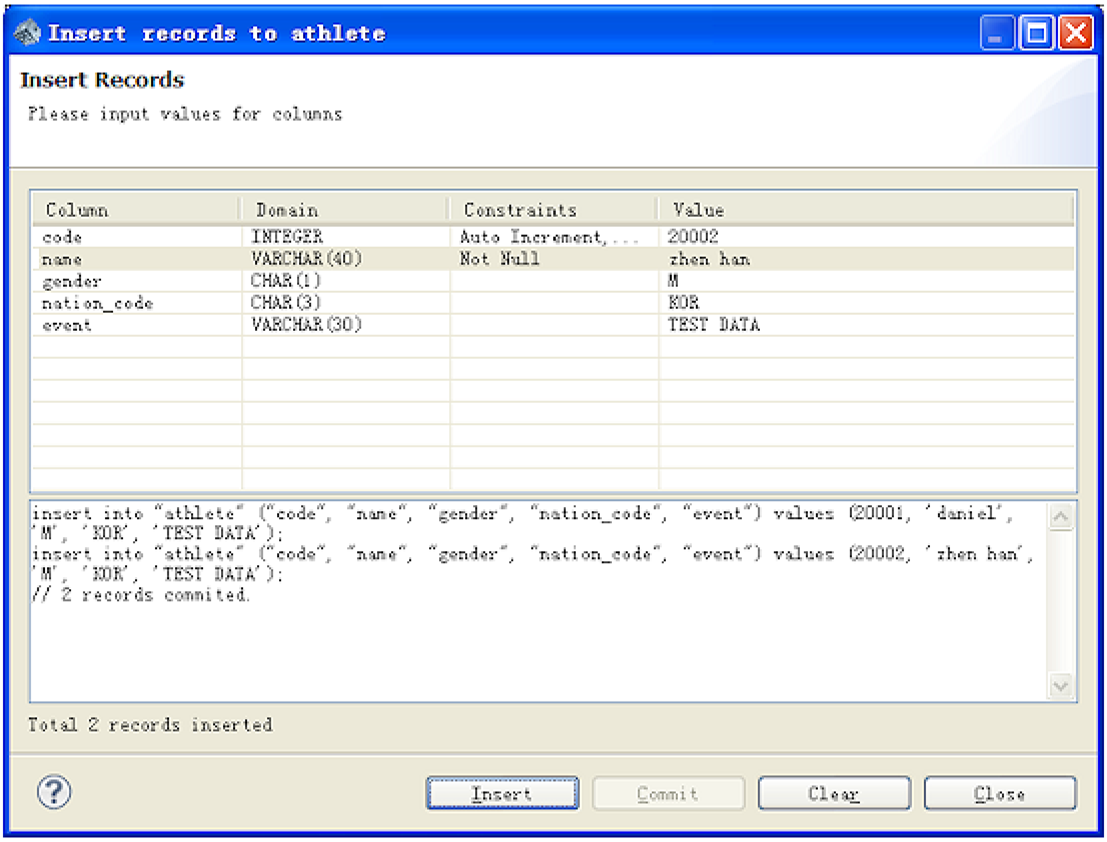
Truncate Table

[More]→[Truncate Table] will delete all data of the table. It can delete all records including indexes and constraints in a table, so it is faster than **[DELETE ALL]**. The **ON DELETE** trigger is not activated when you use the **[TRUNCATE TABLE]**. It is as same as the following syntax.

TRUNCATE TABLE [table-name]

Insert Simple Data

You can insert values for each column while checking its type and constraints.



Column	Domain	Constraints	Value
code	INTEGER	Auto Increment,...	20002
name	VARCHAR(40)	Not Null	zhen han
gender	CHAR(1)		M
nation_code	CHAR(3)		KOR
event	VARCHAR(30)		TEST DATA

```
insert into "athlete" ("code", "name", "gender", "nation_code", "event") values (20001, 'daniel',
'M', 'KOR', 'TEST DATA');
insert into "athlete" ("code", "name", "gender", "nation_code", "event") values (20002, 'zhen han',
'M', 'KOR', 'TEST DATA');
// 2 records commited.
```

Total 2 records inserted

When you add more than one instance, you can separate them by adding a line break character between each query and its result.

You can move the cursor to the next field by pressing the [Enter] key when you entered a value for previous field.

When the [Clear] button is clicked, the value in the input box and the execution history will be initialized.

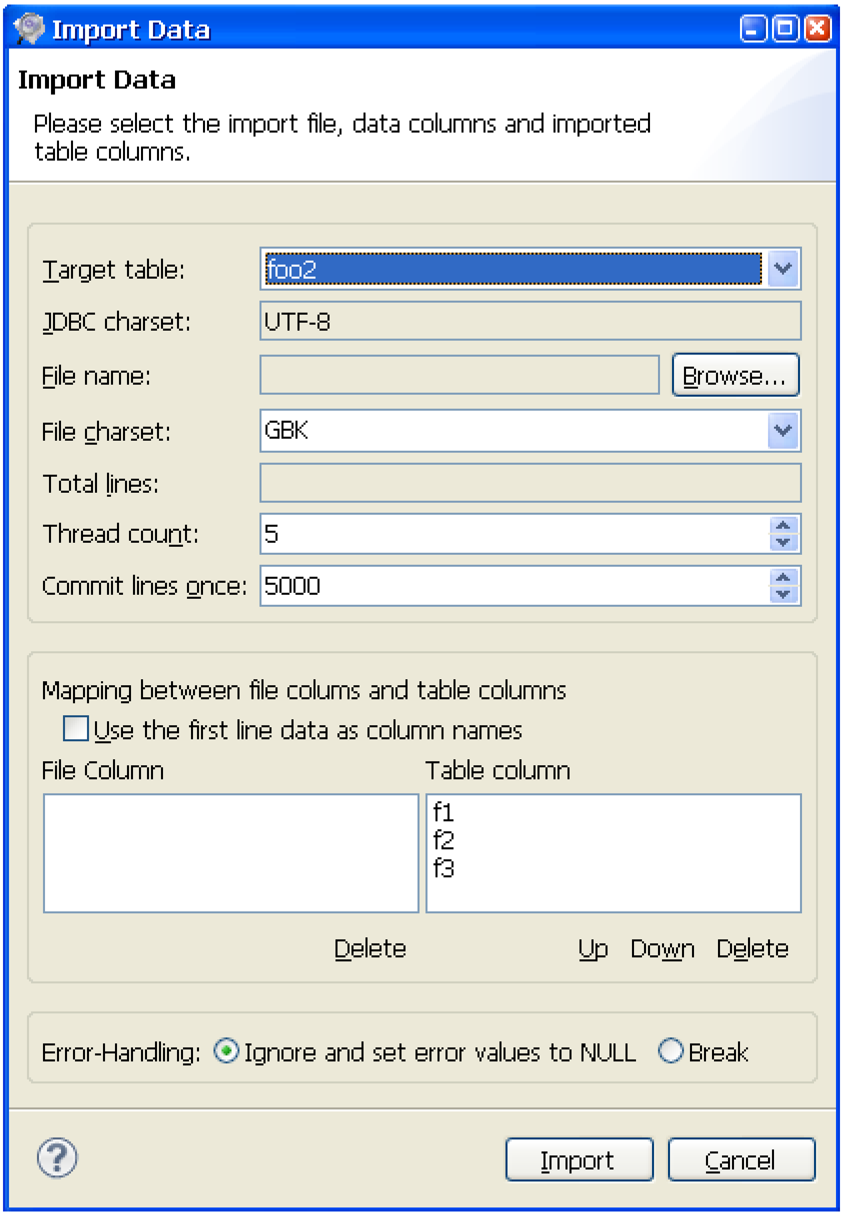
The execution history panel cannot be edited.

For **DATE**, **TIME**, **TIMESTAMP** and **DATETIME** data types, you can enter different data for each type. For example, for a **DATE** type, you can enter data such as **SYSDATE**, **SYS_DATE**, **CURRENT_DATE** and **DATE'***2009-07-05***'**.

Import Data

[Import data] can import data from an Excel or a CSV file into current selected table of the database. Use **[File charset]** to specify the charset when getting data from a file, and use **[JDBC charset]** to specify charset of the data to be imported into the database.

The **[JDBC charset]** only can be changed at the **[Open Connection]** dialog, and the new value will work after the database is reconnected.



Import Data

Please select the import file, data columns and imported table columns.

Target table: foo2

JDBC charset: UTF-8

File name:

File charset: GBK

Total lines:

Thread count: 5

Commit lines once: 5000

Mapping between file colums and table columns

☐ Use the first line data as column names

File Column

Table column

f1

f2

f3

Error-Handling: ● Ignore and set error values to NULL ○ Break

The import process support multi-thread to make better performance. The **[Thread count]** is used to specify how many thread will run during the importing, 5 is recommended and don't make it larger than 10.

[Commit lines once] is used to specify how many rows that memory cache stored in one thread, if the memory cache is full, all data in the cache will be committed to the database and new data will start to fill the memory cache.

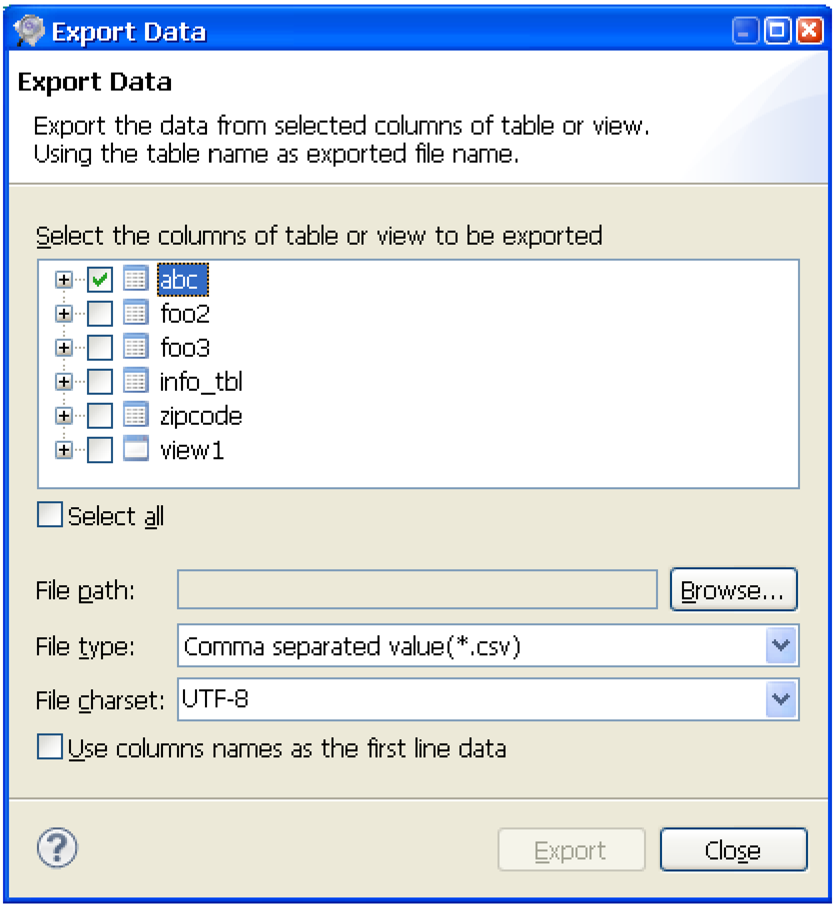
If the first row of the import data is columns' names, the **[Use the first line data as column names]** should be checked.

The buttons [Delete](of file column), [Up], [Down], [Delete](of Table column) are used to specify the correspondence between file columns and table columns.

[Error-Handling] is used to specify what the import progress will do when errors occurred. If **[Ignore]** is checked, the import progress will ignore all errors and show an error report dialog when progress is finished and errors occurred. And if the **[Break]** is checked, the progress will be broke when error occurred, but the committed data will not be rollback.

Export Data

Data (one or more tables) can be exported in Excel (.xls), CSV, SQL, or CUBRID load (.obs) format. The export progress will create a new file for each table and save the files into the path which users specified in **[File path]**. **[File charset]** is used to specify the file charset of the exported data.



Export Data

Export Data

Export the data from selected columns of table or view.
Using the table name as exported file name.

Select the columns of table or view to be exported

- ☑ abc
- □ foo2
- □ foo3
- □ info_tbl
- □ zipcode
- □ view1

□ Select all

File path:

File type: Comma separated value(*.csv)

File charset: UTF-8

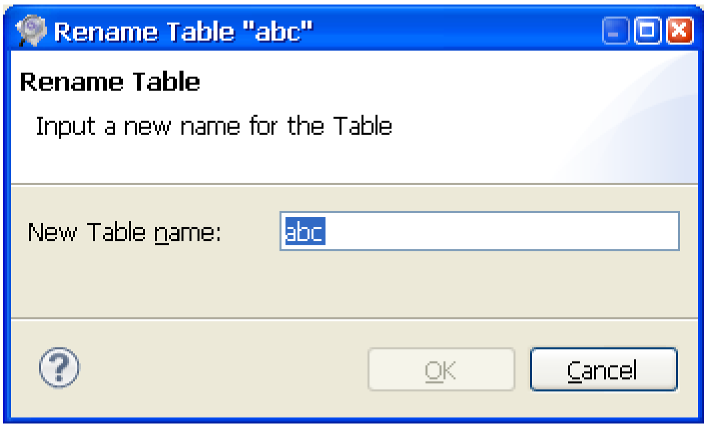
□ Use columns names as the first line data

Drop Table

Drop the selected table. This is the same as the **DROP TABLE** statement.

Rename Table

[More]→[Rename table] is used to change the name of the current table. It is the same as **RENAME TABLE** statement.



Rename Table "abc"

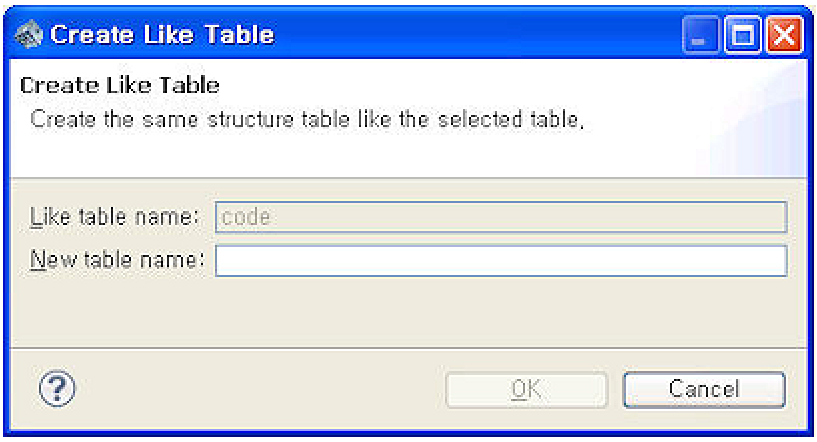
Rename Table

Input a new name for the Table

New Table name: abc

Create Table Like

Create an empty table that has a same schema with an existed one. It works as same as **CREATE TABLE LIKE** statement. For more information, CREATE TABLE LIKE.



Create Like Table

Create Like Table

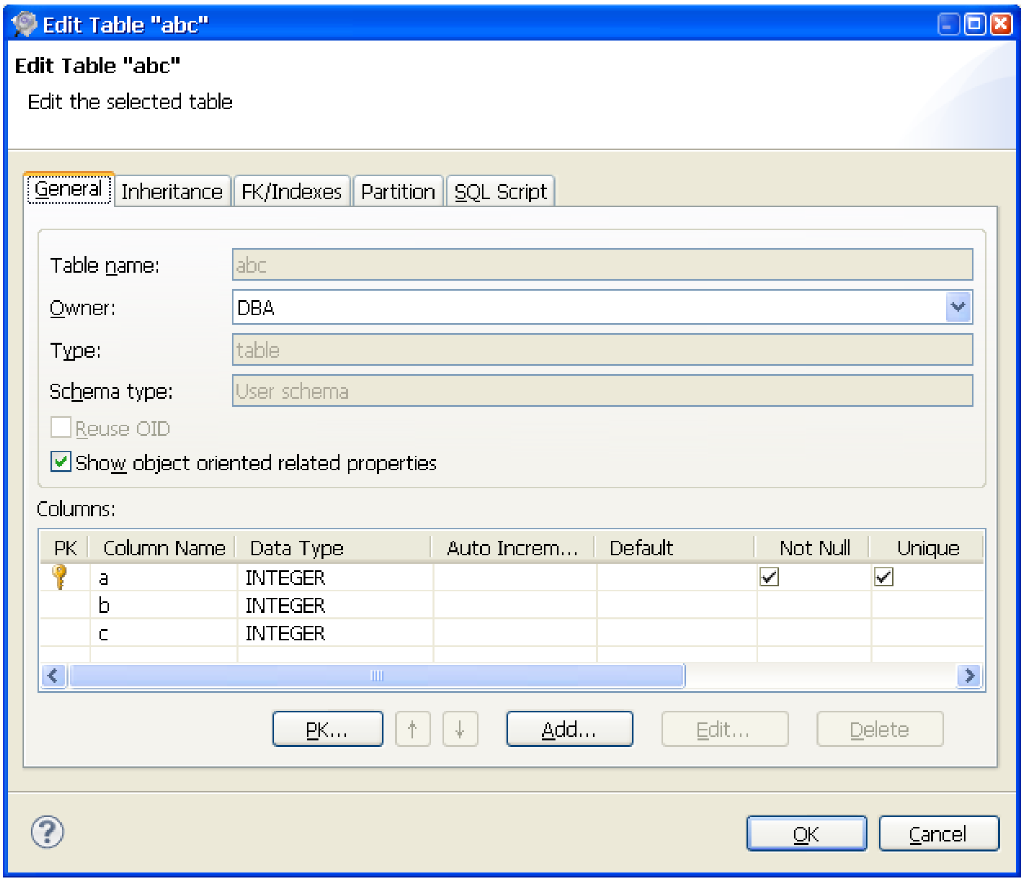
Create the same structure table like the selected table.

Like table name: code

New table name:

Edit Table

[Edit Table] is used to change the table's owner, columns, inheritance, indexes, partition and etc. But the table's name and columns' order can't be changed.



Edit Table "abc"

Edit Table "abc"

Edit the selected table

Table name: abc

Owner: DBA

Type: table

Schema type: User schema

□ Reuse OID

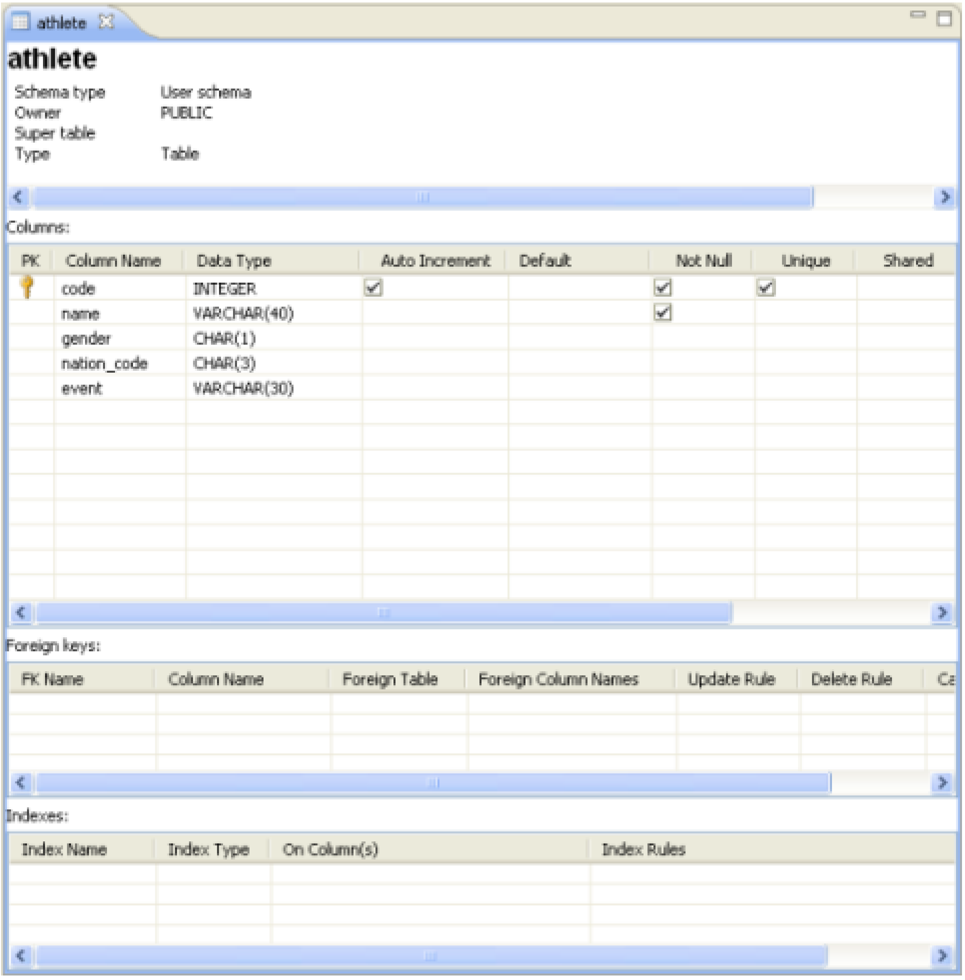
☑ Show object oriented related properties

Columns:

PK	Column Name	Data Type	Auto Increm...	Default	Not Null	Unique
🔑	a	INTEGER			☑	☑
	b	INTEGER				
	c	INTEGER				

Show Schema Information

Double-click the table node or select the [Show Schema Info] menu to view the schema information of the current selected table.



athlete

athlete

Schema type	User schema
Owner	PUBLIC
Super table	
Type	Table

Columns:

PK	Column Name	Data Type	Auto Increment	Default	Not Null	Unique	Shared
🔑	code	INTEGER	☑		☑	☑	
	name	VARCHAR(40)			☑		
	gender	CHAR(1)					
	nation_code	CHAR(3)					
	event	VARCHAR(30)					

Foreign keys:

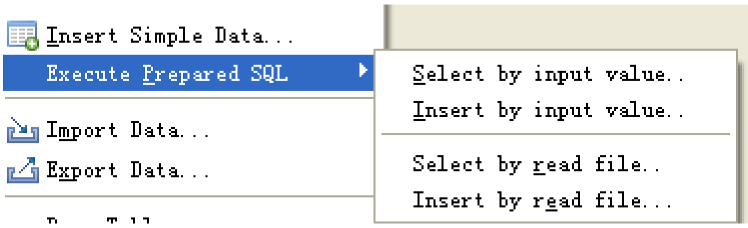
FK Name	Column Name	Foreign Table	Foreign Column Names	Update Rule	Delete Rule	Ca

Indexes:

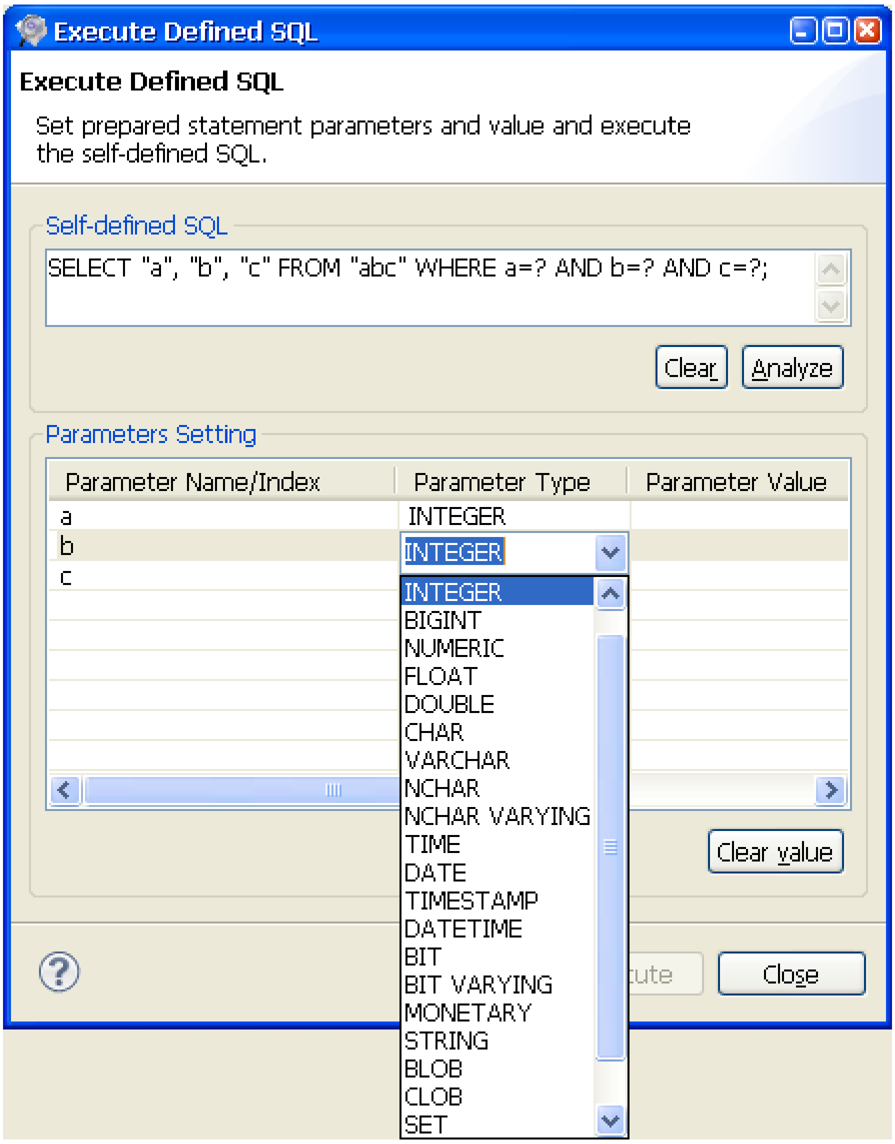
Index Name	Index Type	On Column(s)	Index Rules

Execute Prepared SQL

With **[Execute Prepared SQL]** menu, you can execute "prepared statement," pre-saving a specific query statement and specifying a new parameter value for the statement whenever the query is executed.



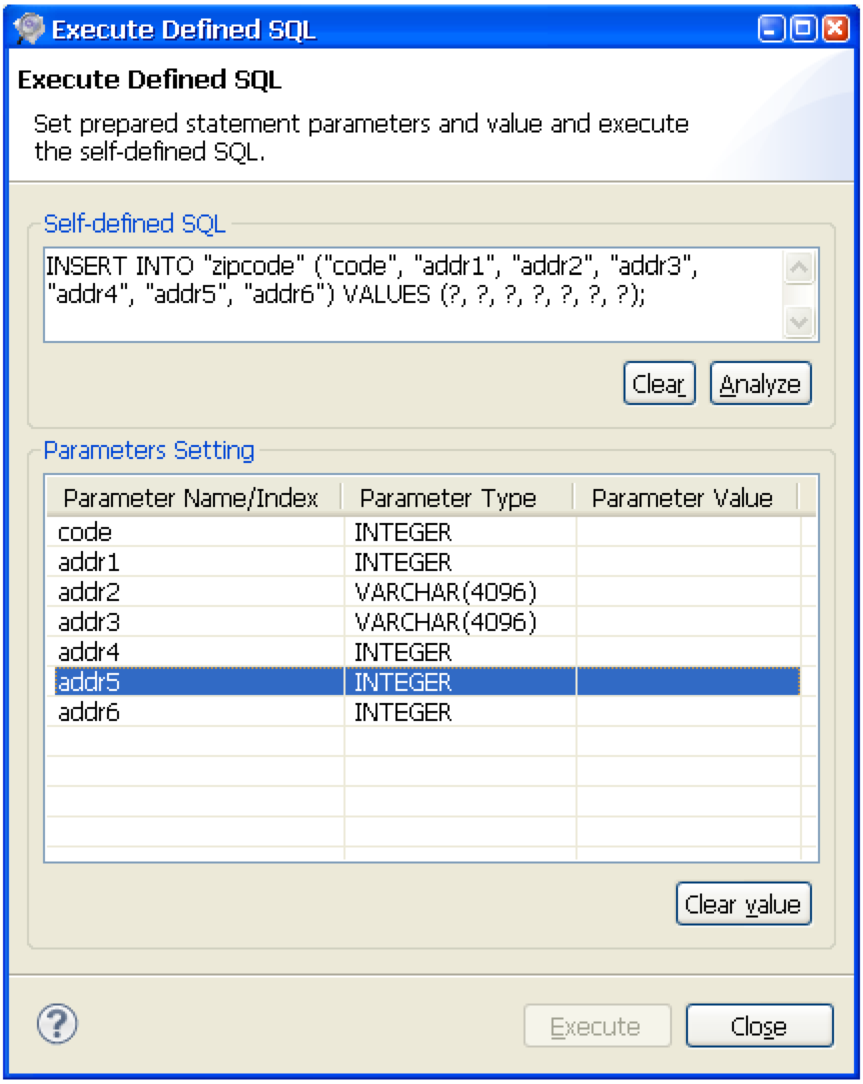
You can execute the **SELECT** statement which meets specified conditions by entering a parameter value of defined "prepared statement" by using **[Select by input value]**.



SELECT "a", "b", "c" FROM "abc" WHERE a=? AND b=? AND c=?;

Parameter Name/Index	Parameter Type	Parameter Value
a	INTEGER	
b	INTEGER	
c		

You can execute the **INSERT** statement by entering a parameter value of defined "prepared statement" by using **[Insert by input value]**.



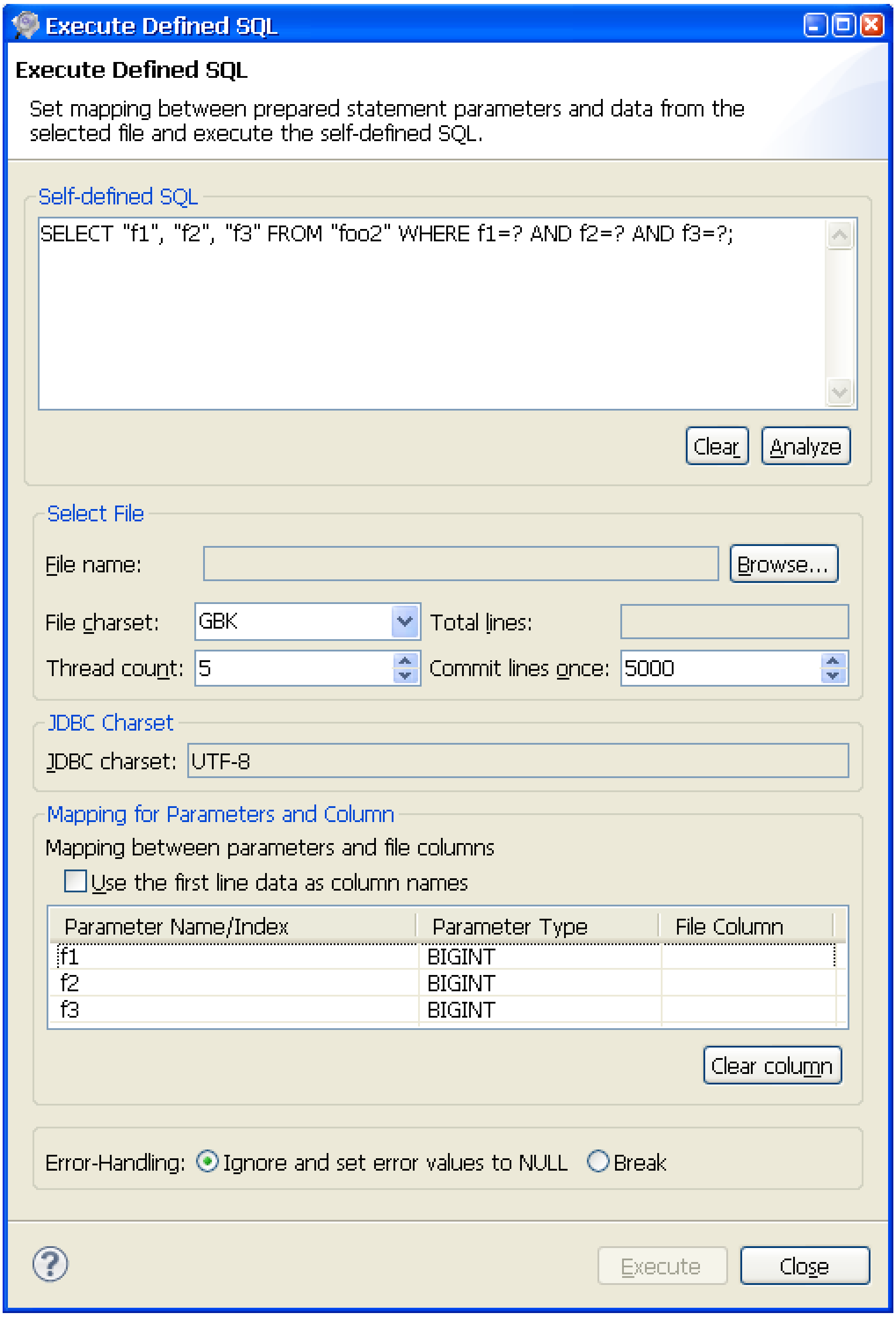
Execute Defined SQL

Set prepared statement parameters and value and execute the self-defined SQL.

```
INSERT INTO "zipcode" ("code", "addr1", "addr2", "addr3", "addr4", "addr5", "addr6") VALUES (?, ?, ?, ?, ?, ?, ?);
```

Parameter Name/Index	Parameter Type	Parameter Value
code	INTEGER	
addr1	INTEGER	
addr2	VARCHAR(4096)	
addr3	VARCHAR(4096)	
addr4	INTEGER	
addr5	INTEGER	
addr6	INTEGER	

You can use **[Select by read file]** when repeating the execution of **SELECT** statement by inputting several parameter values in the defined "prepared statement." The data is saved in Excel (.xls) or CSV format. You can configure a file charset where parameter values are saved, the number of concurrent threads to be executed, and commit cycle.



Execute Defined SQL

Set mapping between prepared statement parameters and data from the selected file and execute the self-defined SQL.

Self-defined SQL

```
SELECT "f1", "f2", "f3" FROM "foo2" WHERE f1=? AND f2=? AND f3=?;
```

File name:

File charset: GBK

Total lines:

Thread count: 5

Commit lines once: 5000

JDBC charset: UTF-8

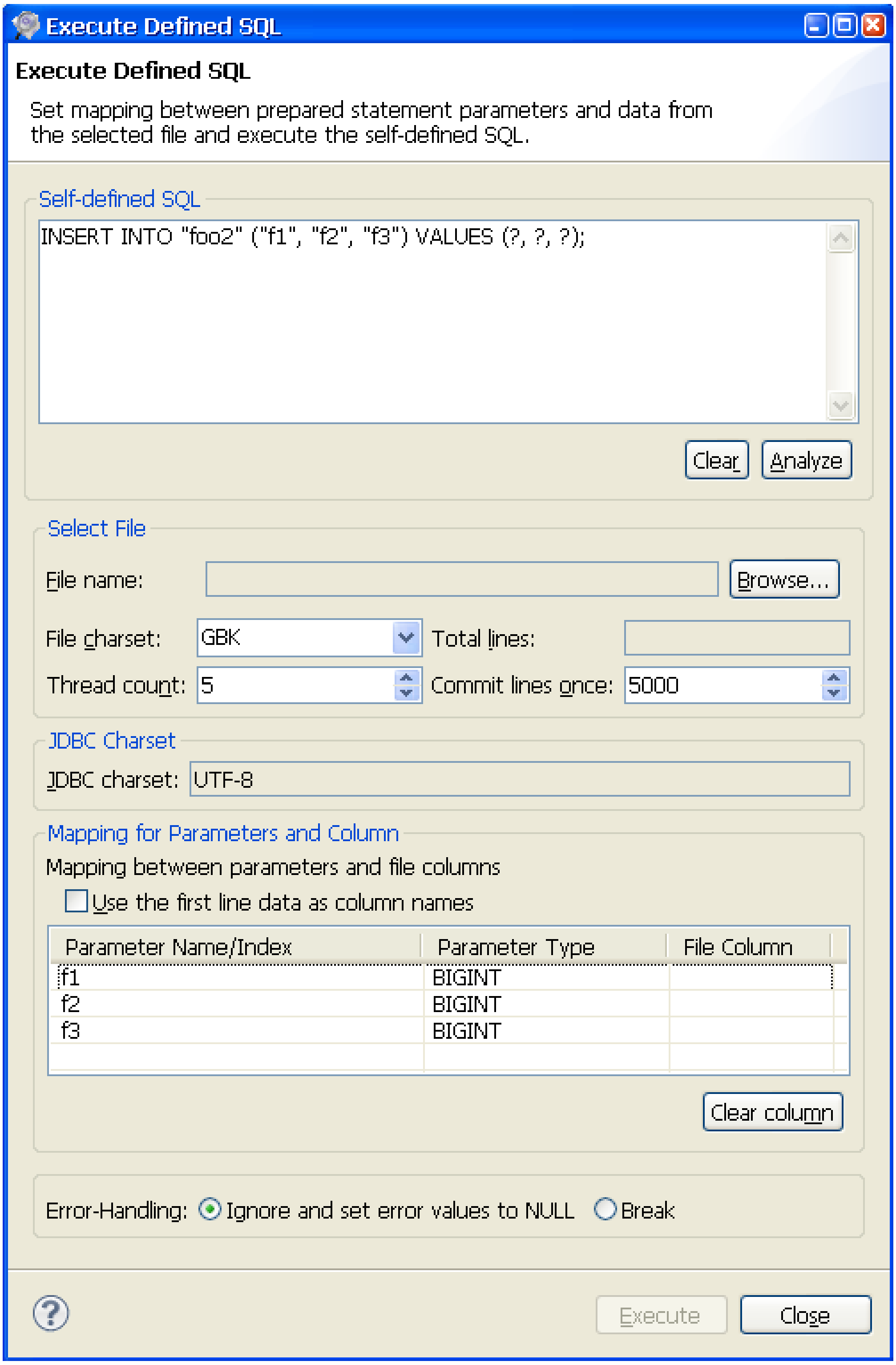
Mapping between parameters and file columns

□ Use the first line data as column names

Parameter Name/Index	Parameter Type	File Column
f1	BIGINT	
f2	BIGINT	
f3	BIGINT	

Error-Handling: ● Ignore and set error values to NULL ○ Break

You can use **[Insert by read file]** when repeating the execution of **INSERT** statement by inputting several values in the defined "prepared statement." The data is saved in Excel (.xls) or CSV format. You can configure a file charset where parameter values are saved, the number of concurrent threads to be executed, and commit cycle. **[JDBC charset]** can be changed in **[Open connection]** dialog, and it works when the database is reconnected.



Execute Defined SQL

Set mapping between prepared statement parameters and data from the selected file and execute the self-defined SQL.

Self-defined SQL

```
INSERT INTO "foo2" ("f1", "f2", "f3") VALUES (?, ?, ?);
```

Select File

File name:

File charset: GBK

Total lines:

Thread count: 5

Commit lines once: 5000

JDBC Charset

JDBC charset: UTF-8

Mapping for Parameters and Column

Mapping between parameters and file columns

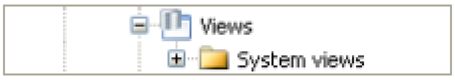
□ Use the first line data as column names

Parameter Name/Index	Parameter Type	File Column
f1	BIGINT	
f2	BIGINT	
f3	BIGINT	

Error-Handling: ● Ignore and set error values to NULL ○ Break

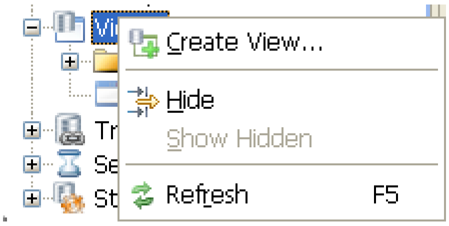
View

When the database is connected, views and system views that are authorized to current login user will be listed in the navigation tree.

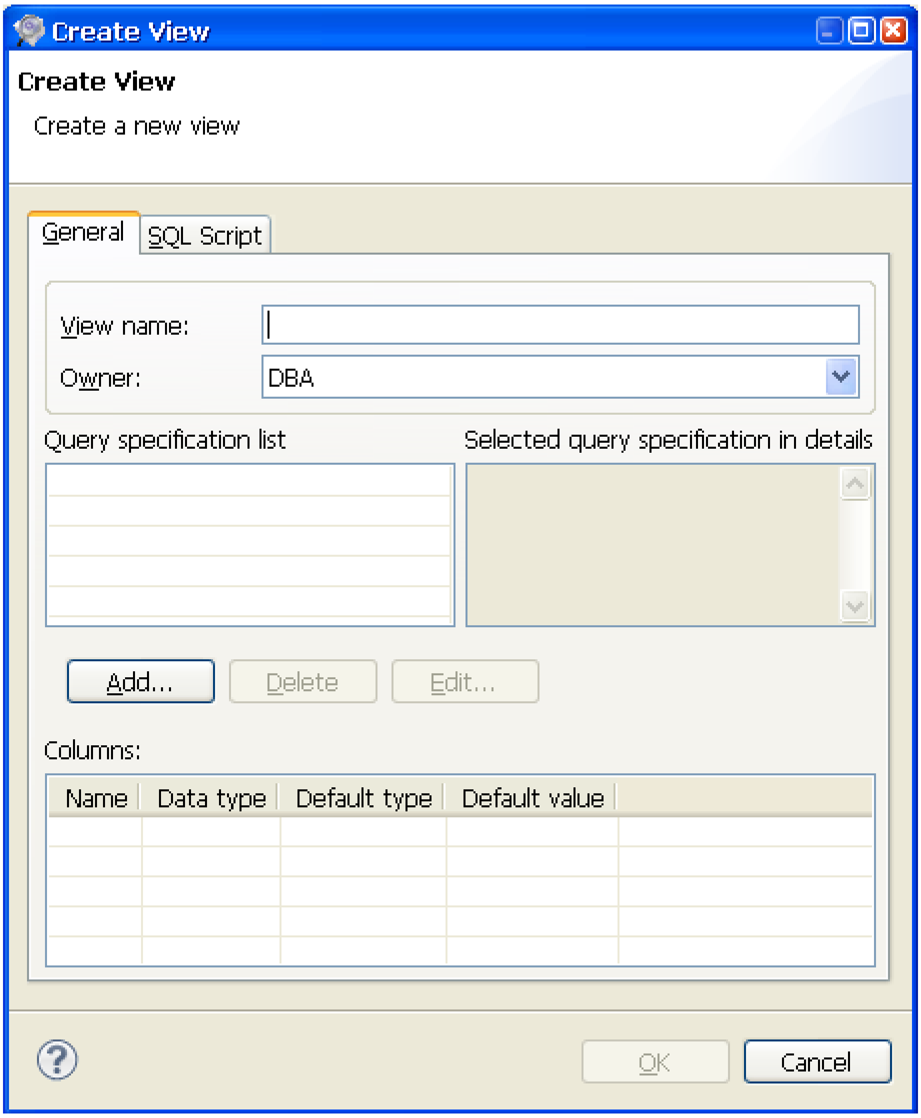


Create View

Right-click [Views] in the navigation tree and then select [Create View].



Specify the view’s name and the owner and click the **[Add]** button, and then enter the query statement for the view to be created. The SQL statement of create view will be generated in the [SQL Script] tab.



Create View

Create View

Create a new view

View name:

Owner: DBA

Name	Data type	Default type	Default value

Select All

The context menu **[Select All]** of view node will open a new query editor and select all data of current selected view. The view node also can be dragged and dropped into a query editor to generate a select statement of the view.

Select Count

Retrieves a total row count of the table and it works as same as the following syntax.

```
SELECT COUNT (*) FROM [view]
```

Export Data

You can export all data of the view to an Excel, CSV, SQL or CUBRID load format. For more please refer to the **[Export data]** of table.

Drop View

Drop the selected view. This is same as the **DROP VIEW** statement.

Rename View

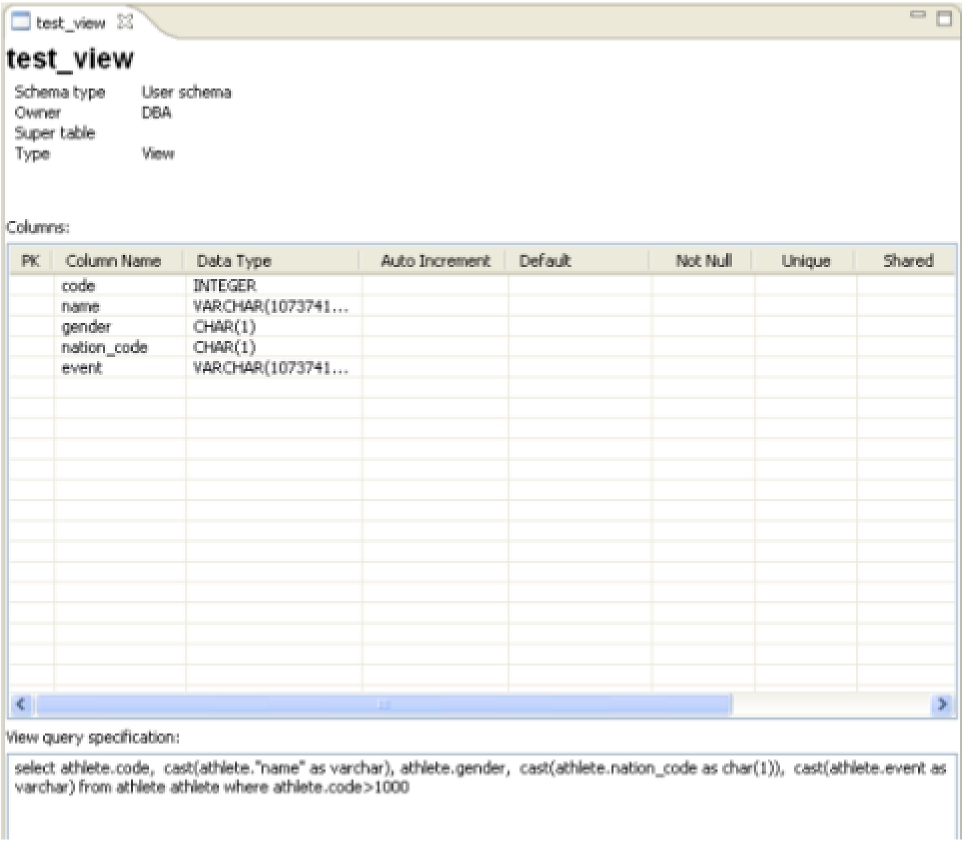
You can change the name of the current view. This is as same as the **RENAME VIEW** statement.

Edit View

You can use all functionalities of [Create View] in [Edit View] as well.

View Information

You can check the schema information of the view by double-clicking it.



test_view

Schema type	User schema
Owner	DBA
Super table	
Type	View

Columns:

PK	Column Name	Data Type	Auto Increment	Default	Not Null	Unique	Shared
	code	INTEGER					
	name	VARCHAR(1073741...					
	gender	CHAR(1)					
	nation_code	CHAR(1)					
	event	VARCHAR(1073741...					

View query specification:

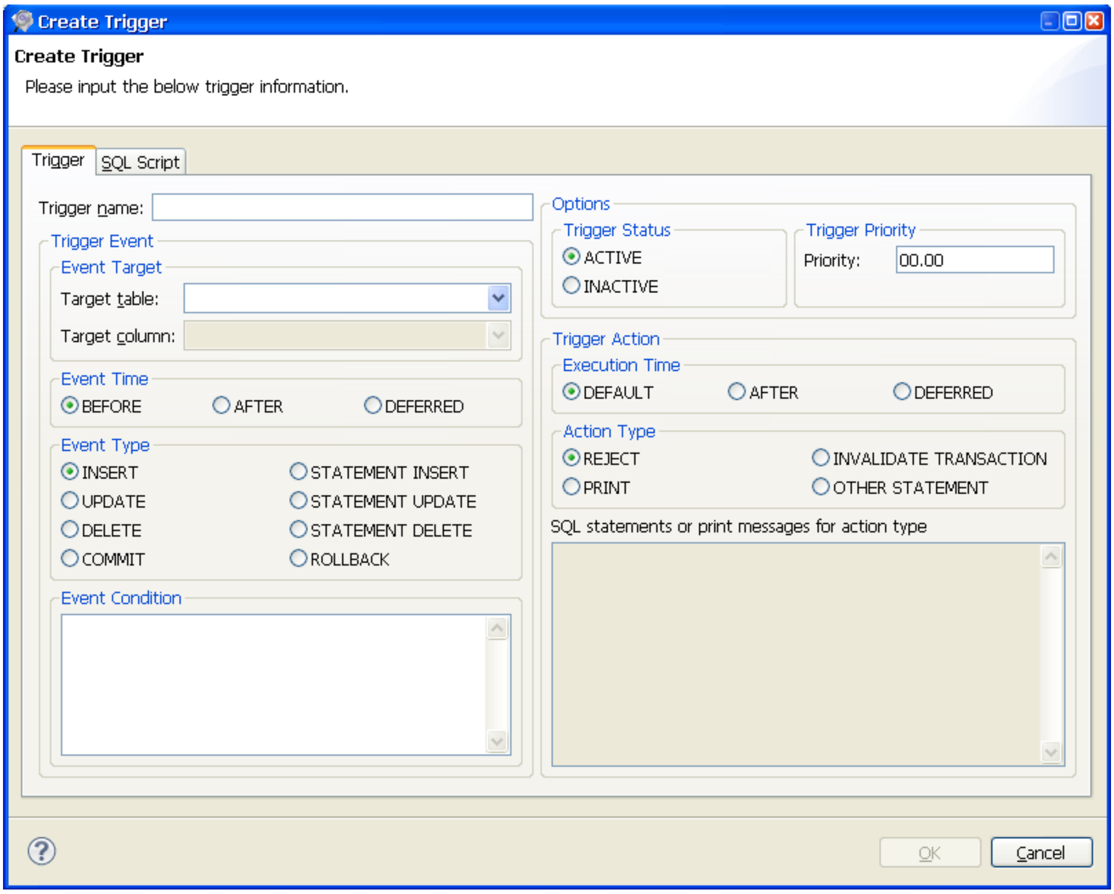
```
select athlete.code,  cast(athlete."name" as varchar), athlete.gender,  cast(athlete.nation_code as char(1)),  cast(athlete.event as
varchar) from athlete athlete where athlete.code>1000
```

Trigger

When the database is connected, triggers that are authorized to current login user will be listed in the navigation tree.

Create Trigger

Right-click [Triggers] in the navigation tree and then select **[Create Trigger]**.



Create Trigger

Create Trigger

Please input the below trigger information.

Trigger name:

Trigger Event

Event Target

Target table:

Target column:

Event Time

● BEFORE ○ AFTER ○ DEFERRED

Event Type

● INSERT ○ STATEMENT INSERT
○ UPDATE ○ STATEMENT UPDATE
○ DELETE ○ STATEMENT DELETE
○ COMMIT ○ ROLLBACK

Event Condition

Options

Trigger Status

● ACTIVE
○ INACTIVE

Trigger Priority

Priority: 00.00

Trigger Action

Execution Time

● DEFAULT ○ AFTER ○ DEFERRED

Action Type

● REJECT ○ INVALIDATE TRANSACTION
○ PRINT ○ OTHER STATEMENT

SQL statements or print messages for action type

- **Trigger name**: Enter the name of the trigger to be added.
- **Event Time**: Select the point of time when the condition of the trigger is to be evaluated. You can select from **BEFORE**, **AFTER** and **DEFERRED**.
- **Event Type**: Select the type of the event to be occurred. Event types are **INSERT**, **DELETE**, **UPDATE**, **STATEMENT INSERT**, **STATEMENT DELETE**, **STATEMENT UPDATE**, **COMMIT** and **ROLLBACK**.
- **Target table/Column**: Enter the target table and column information.
- **Event Condition**: Enter the condition for the trigger action.
- **Execution Time**: Specify the point of time when the trigger is to be executed. If **default** is selected, the trigger fires based on the point of time when its condition is validated.
- **Action Type**: Select the type of the trigger action. If the type of the trigger action is **PRINT**, **OTHER** or **STATEMENT**, you can enter additional information in the [SQL statements or print messages] below.

- **Trigger Status**: You can specify whether to activate or deactivate the trigger to be added.
- **Trigger Priority**: You can set the priority of the trigger. The priority value is a **FLOAT** and can be between 00.00 and 9999.99.

Drop Trigger

Right-click the trigger in the navigation tree and then select **[Drop Trigger]**.

Edit Trigger

Right-click the trigger in the navigation tree and then select **[Edit Trigger]**. You can simply edit a value in the **[Priority]** field.

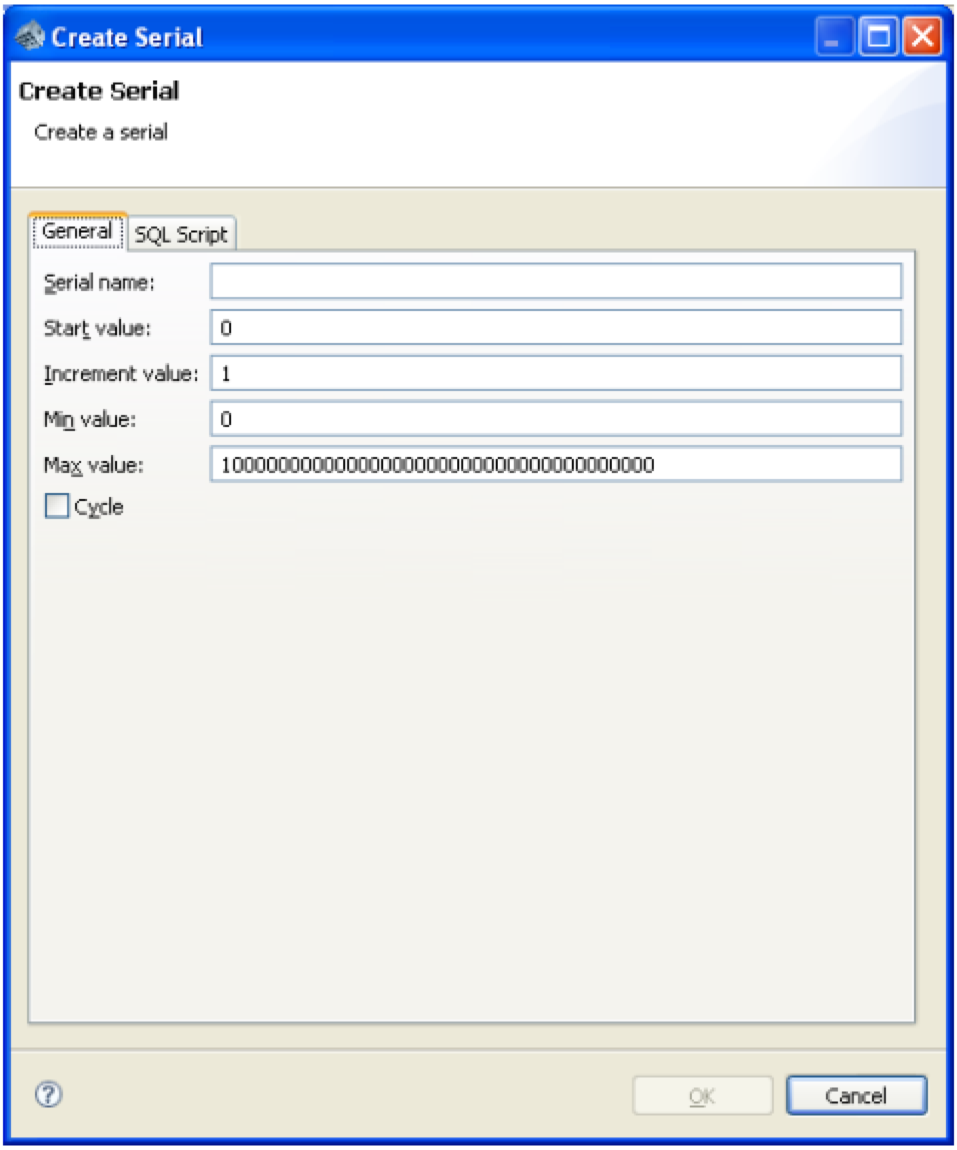
Serial

When the database is connected, serials that are authorized to current login user will be listed in the navigation tree.

Create Serial

Right-click [Serials] in the navigation tree and then select [Create Serial].

The name of the serial must be unique in the database.



Create Serial

Create a serial

Serial name:

Start value: 0

Increment value: 1

Min value: 0

Max value: 10000000000000000000000000000000000000

□ Cycle

Drop Serial

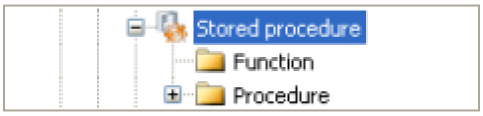
Right-click the serial in the navigation tree and then select [Drop Serial].

Edit Serial

Right-click the serial in the navigation tree and then select [Edit Serial].

Stored Procedure

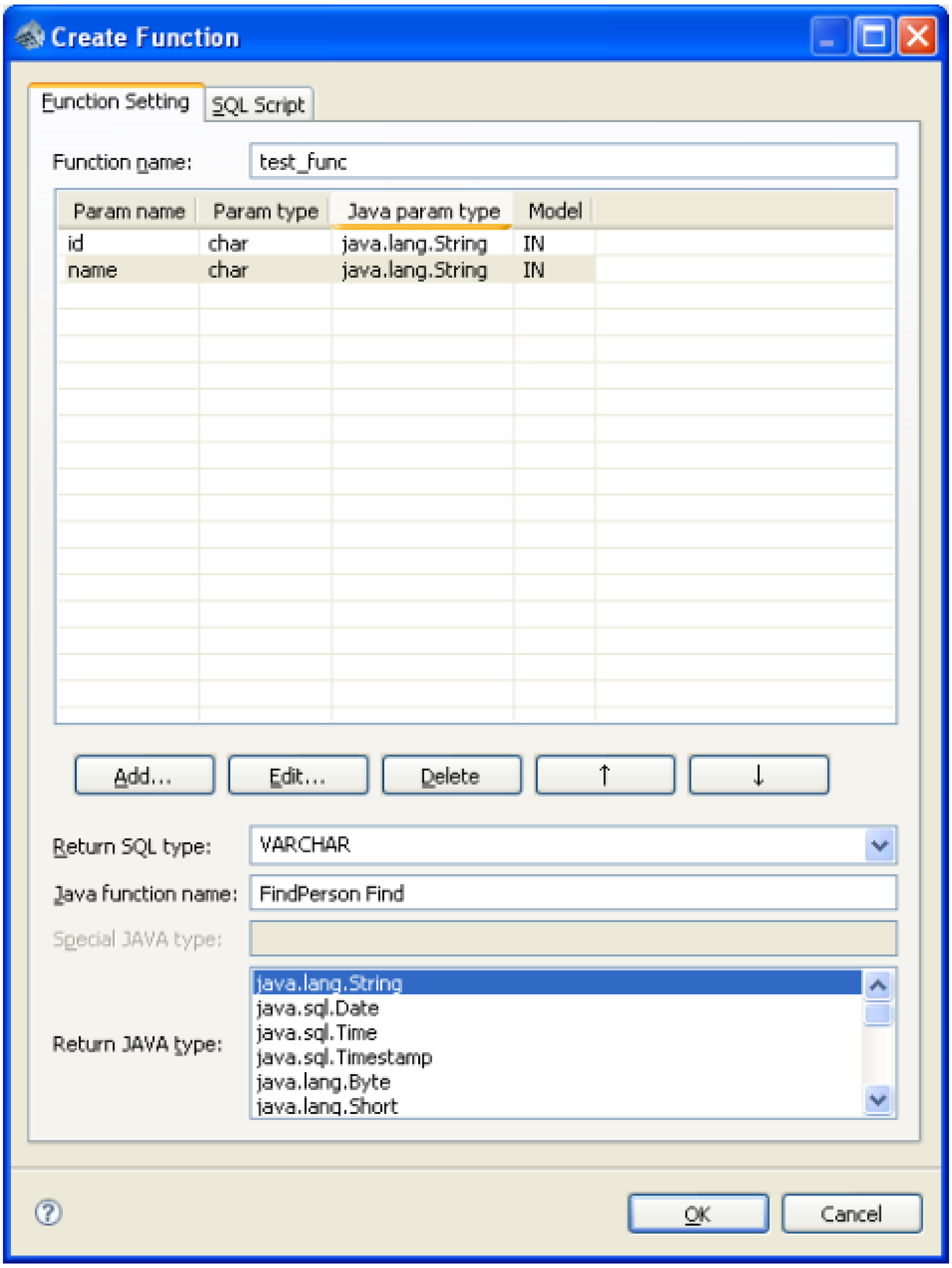
When the database is connected, procedures that are authorized to current login user will be listed in the navigation tree.



Create Function

It is a feature that registers a function written in Java in the database server with the **loadjava** command and adds a database function in order to use the given Java function.

Right-click **[Stored procedure]** in the navigation tree and then select **[Create Function]**.



Function name: test_func

Param name	Param type	Java param type	Model
id	char	java.lang.String	IN
name	char	java.lang.String	IN

Return SQL type: VARCHAR

Java function name: FindPerson Find

Special JAVA type:

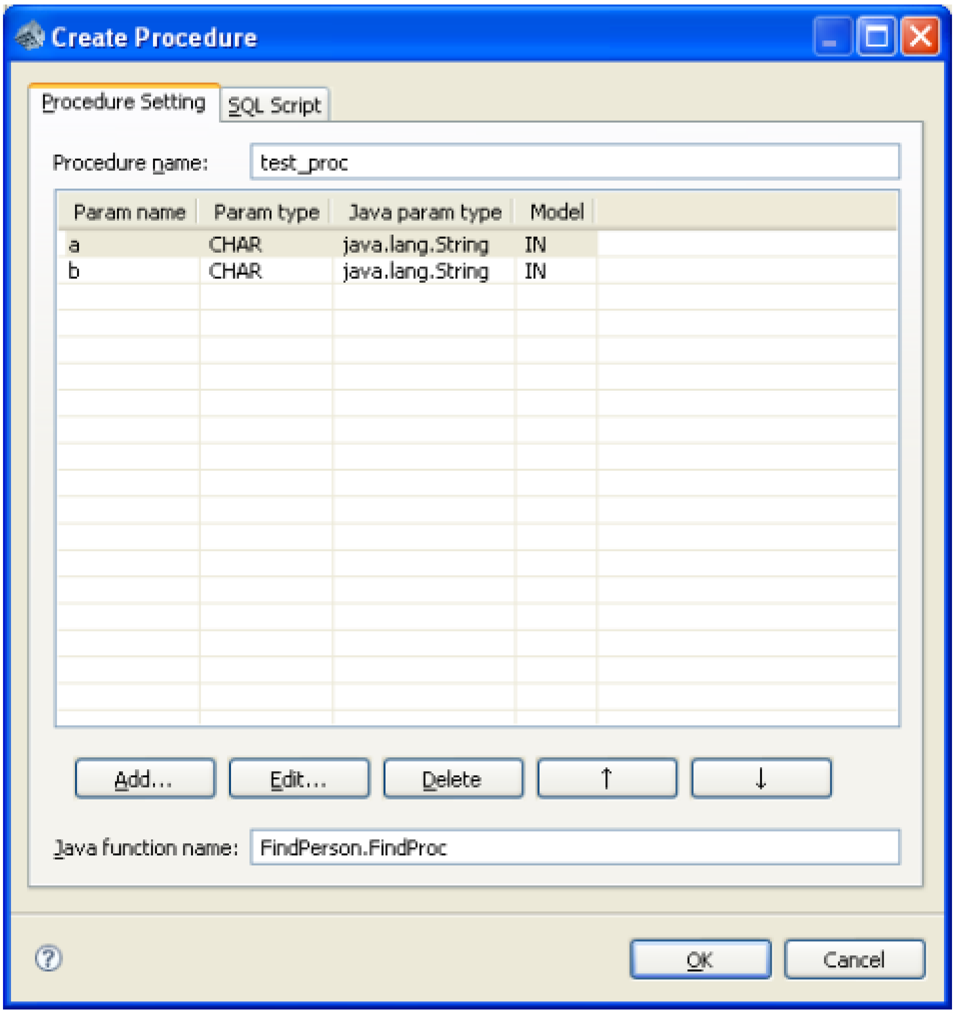
Return JAVA type: java.lang.String, java.sql.Date, java.sql.Time, java.sql.Timestamp, java.lang.Byte, java.lang.Short

Only Java types compatible with the selected SQL type are displayed so that the users can’t make a mistake with type mapping.

Create Procedure

It is a feature that registers a procedure written in Java in the database server with the **loadjava** command and creates a database procedure in order to use the given Java procedure.

Right-click **[Stored procedure]** in the navigation tree and then select **[Create Procedure]**.



Procedure name: test_proc

Param name	Param type	Java param type	Model
a	CHAR	java.lang.String	IN
b	CHAR	java.lang.String	IN

Java function name: FindPerson.FindProc

Only Java types compatible with the selected SQL type are displayed so that the user does not make a mistake with type mapping.

The following is a statement created by using the Create Procedure wizard above.

CREATE PROCEDURE "test_proc"("a" CHAR,"b" CHAR)

AS LANGUAGE JAVA

NAME 'FindPerson.FindProc(java.lang.String,java.lang.Integer)'

Drop Function/Procedure

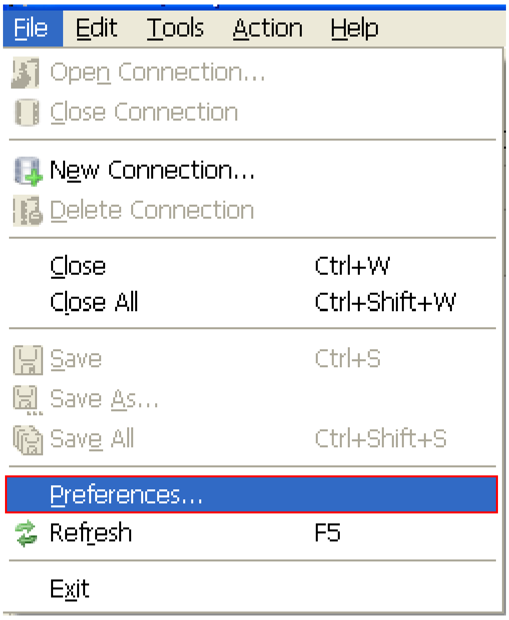
Right-click a certain function or a procedure and then select **[Drop Function]** or **[Drop Procedure]**.

Edit Function/Procedure

Right-click a certain function or a procedure and then select [Edit Function] or [Edit Procedure]. The feature works the same as with Create Function/Procedure.

Preference

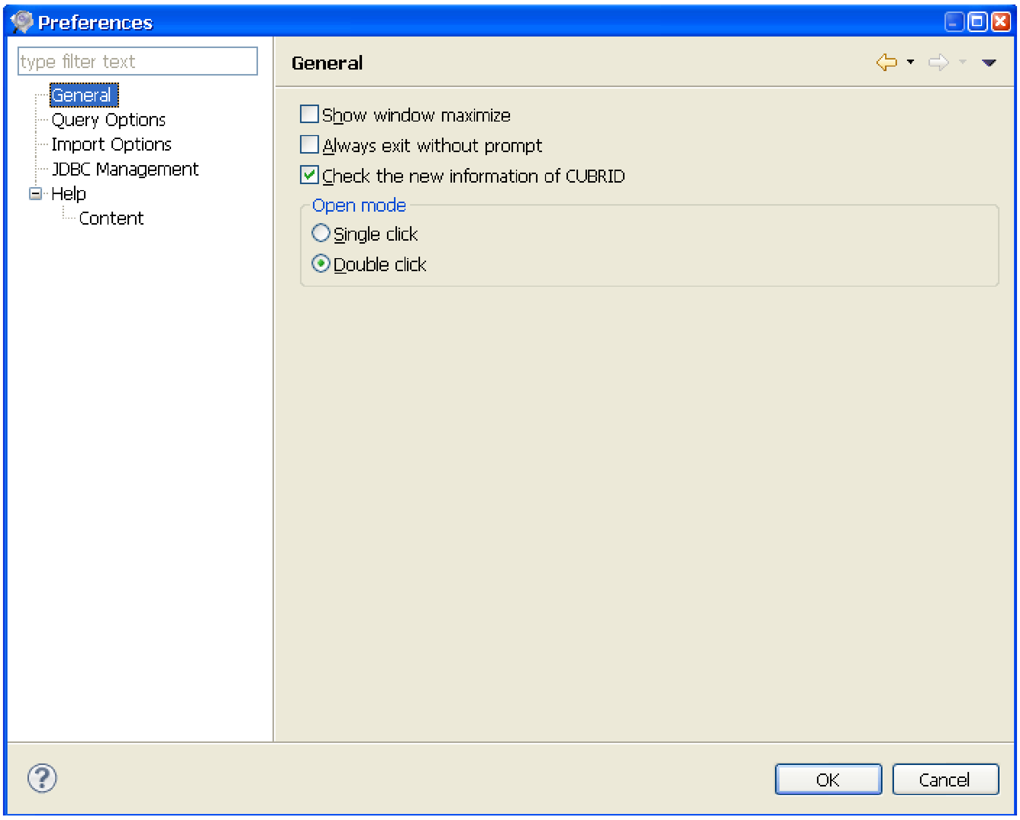
[File]→[Preference] menu is used to open the preference dialog:



Users can change some preference options by using the dialog.

General

This page is used to set up some general options.



General

□ Show window maximize

□ Always exit without prompt

☑ Check the new information of CUBRID

Open mode

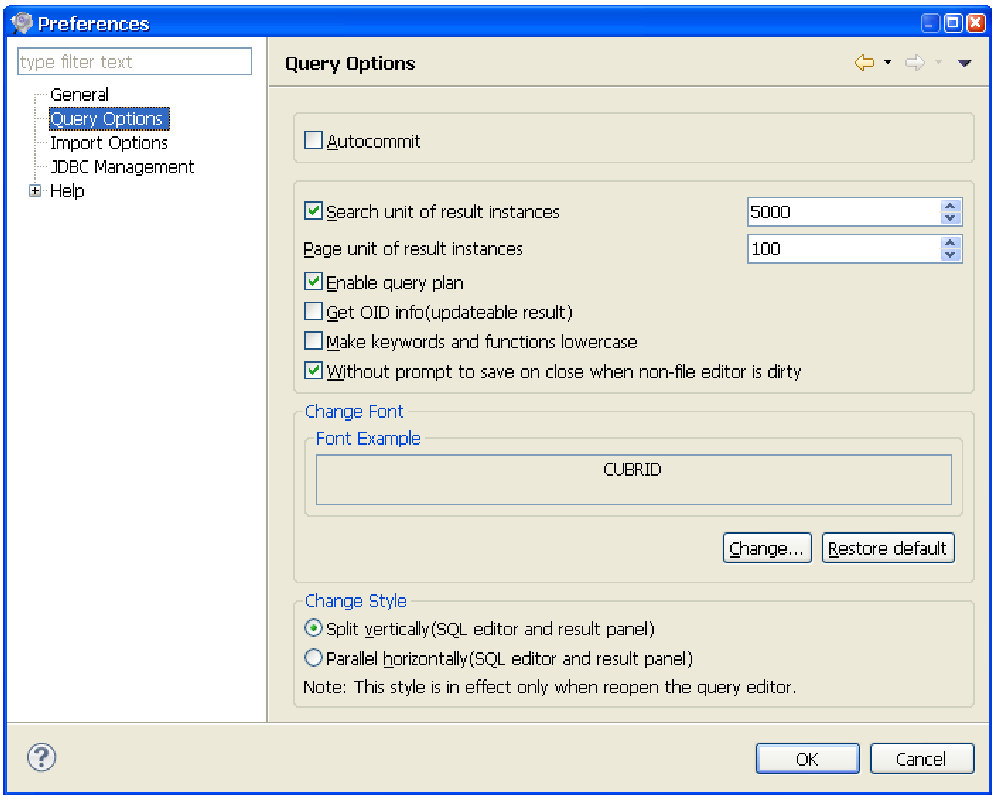
○ Single click

● Double click

- **Show window maximize:** If it is checked, the query browser will be maximized when it started.
- **Always exit without prompt:** if it is checked, there is no prompt when query browser is closing.
- **Check the new information of CUBRID:** if it is checked, an new information of CUBRID editor will be opened when the query browser started.
- **Open mode:** use single click or double dick to open nodes in the connection explorer.

Query Options

This page is used to set some options about query editor.



Query Options

□ Autocommit

☑ Search unit of result instances 5000

Page unit of result instances 100

☑ Enable query plan

□ Get OID info(updateable result)

□ Make keywords and functions lowercase

☑ Without prompt to save on close when non-file editor is dirty

Change Font

Font Example

CUBRID

Change Style

● Split vertically(SQL editor and result panel)

○ Parallel horizontally(SQL editor and result panel)

Note: This style is in effect only when reopen the query editor.

- **Autocommit**: You can set a default value so that autocommit is performed after a query is executed in the query edit pane. Even when [Autocommit] is selected, you can set ( )/clear ( ) the function from the toolbar in the Query Editor.
- **Search unit of result instances**: Sets the number of result rows to be fetched at once from the database after the execution of the query. That is, if the number of search result rows is 7,000 when the value is set to 5,000, you can select whether or not to continue to fetch more search results after 5,000 rows are fetched. In addition, if you execute queries that satisfy the following conditions, **ROWNUM** is automatically added based on the set value to prevent using excessive server resources.

  WHERE condition is missing.

  GROUP BY is not used.

  ORDER BY is not used.

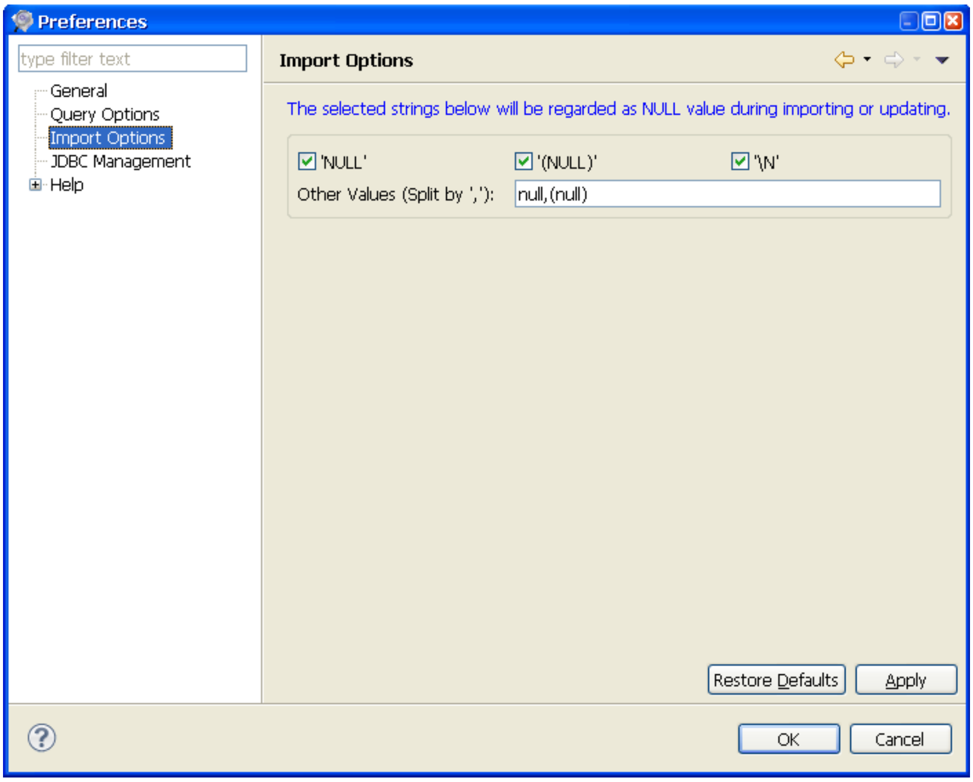
  Aggregate functions (**SUM**, **COUNT**, **MIN**, **MAX**, **AVG**, **STDDEV** and **VARIANCE**) are not used.
- Hierarchical Query is not used.
- **Page unit of result instances**: You can navigate in the query results pane by paging (|⇦ ⇧ ⇩ ⇨|) specified number of records.
- **Enable query plan** : You can check the query plan before and after its execution. Selecting this option may slightly affect the query execution time because plan information will be created in advance for queries to be executed.

- **Get OID info**: Fetches OID information during the query execution. This allows you to directly modify/delete data in the query results pane. However, "NONE" is displayed if the OID cannot be fetched as with Join queries. Selecting this option may extend the query execution time.
- **Make keywords and functions lowercase**: The auto-complete will show the keywords as lowercase.
- **Without prompt to save on close when non-file editor is dirty**: If the editor doesn't open a file, the save prompt will not pop on close whether the editor is dirty or not.
- **Change font & size**: You can change fonts and their sizes to be used in the Query Editor.
- **Change Style**: Change the show style of query editor's editor panel and result panel.

Import Options

Import options is used to specify which value will be turn to NULL when import data to database.

Strings 'NULL', '(NULL)' and '\N' are the default options and users can input some custom values into the text area, the max length of text is 256.



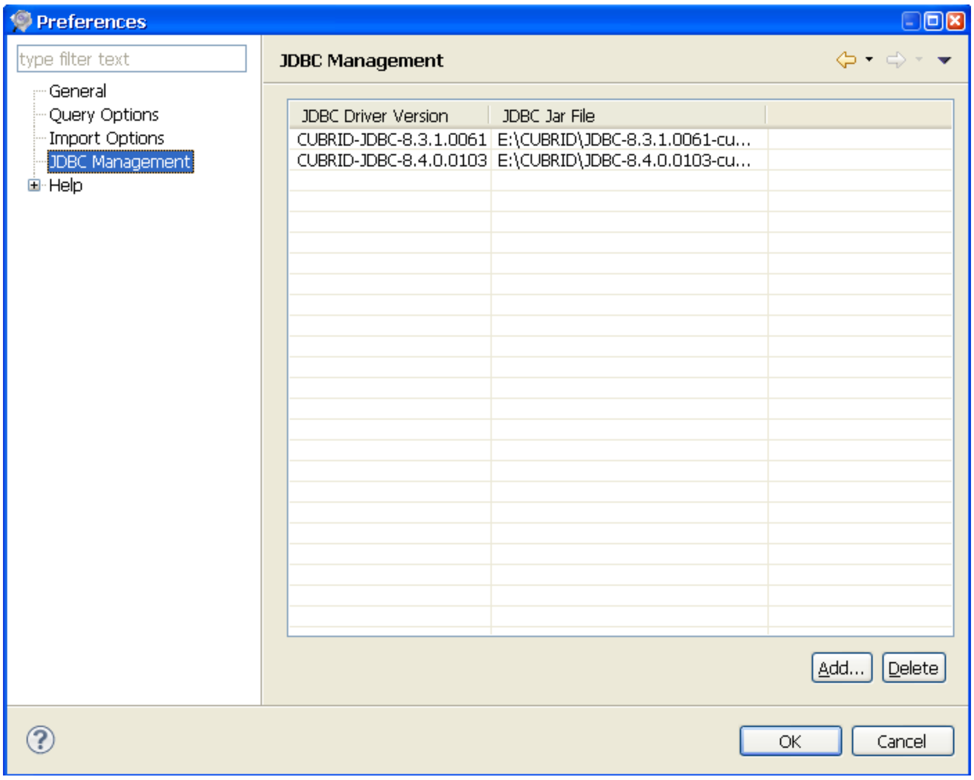
The selected strings below will be regarded as NULL value during importing or updating.

☑ 'NULL' ☑ '(NULL)' ☑ '\N'

Other Values (Split by ','): null,(null)

JDBC Management

Users can manage various CUBRID JDBC drivers by this page.



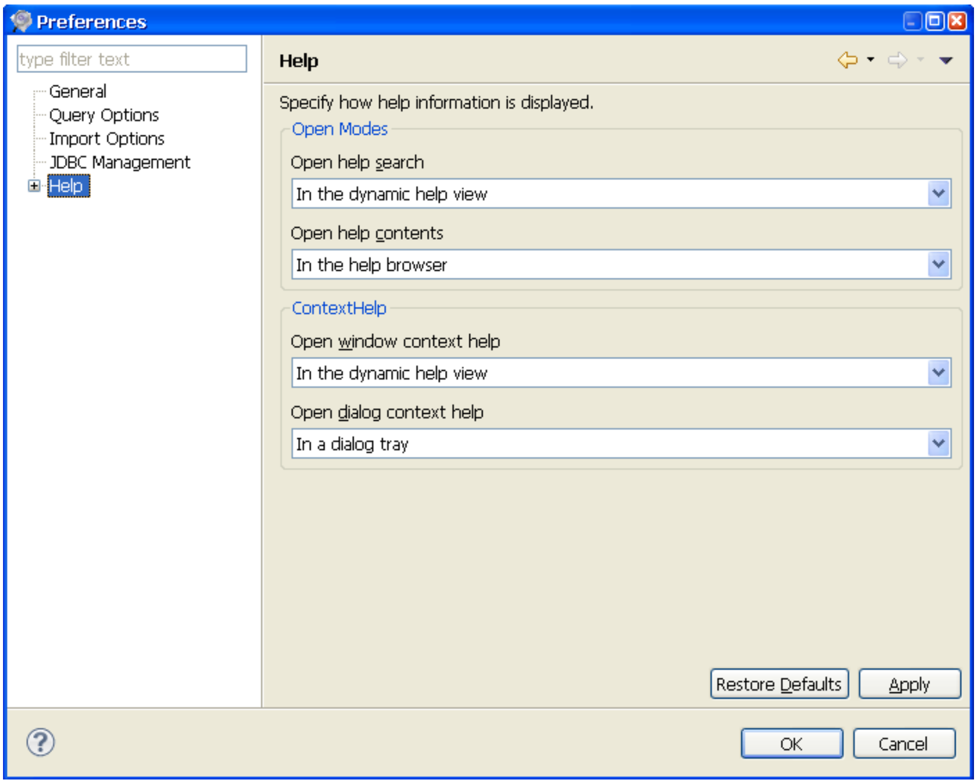
JDBC Driver Version	JDBC Jar File
CUBRID-JDBC-8.3.1.0061	E:\CUBRID\JDBC-8.3.1.0061-cu...
CUBRID-JDBC-8.4.0.0103	E:\CUBRID\JDBC-8.4.0.0103-cu...

Button **[Add]** will open a file selection dialog, and the specified CUBRID JDBC driver file will be added to the JDBC driver list.

Button **[Delete]** will delete current selected JDBC driver from the list.

Help

Set up preference options of CUBRID Query browser help:



Help

Specify how help information is displayed.

Open Modes

Open help search: In the dynamic help view

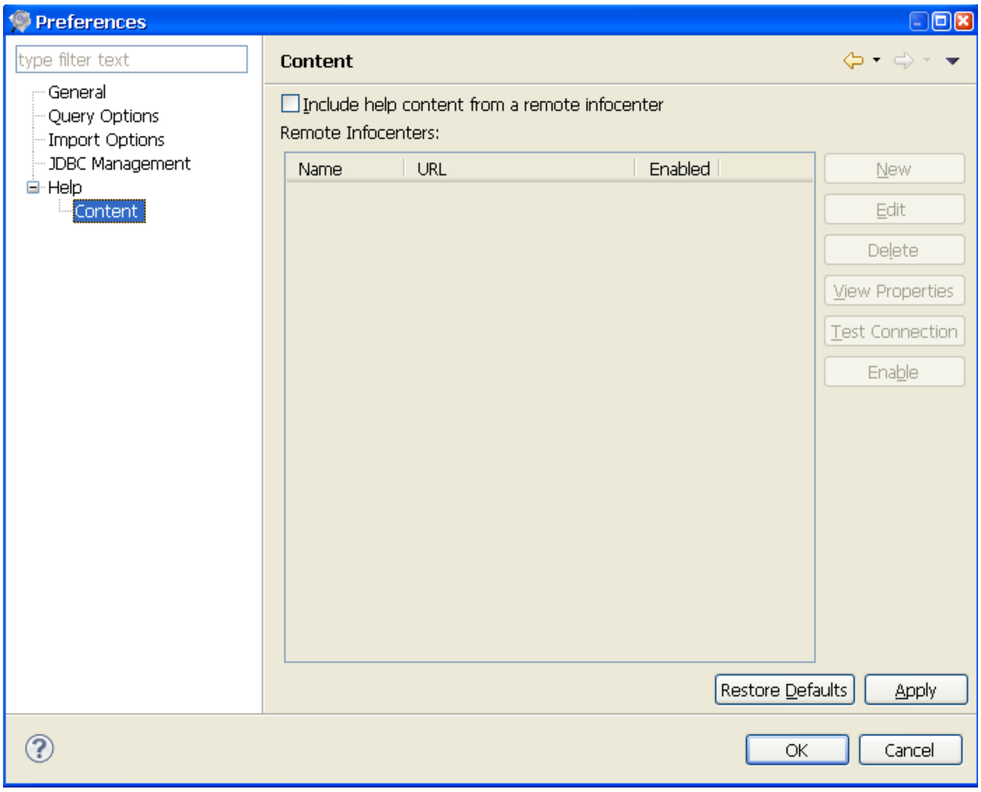
Open help contents: In the help browser

ContextHelp

Open window context help: In the dynamic help view

Open dialog context help: In a dialog tray

Set up the remove help content:



Content

☐ Include help content from a remote infocenter

Remote Infocenters:

Name	URL	Enabled