



Planche

for MySQL

정 주원

정 주원




NIPA Global Open Frontier Lab

Korea Sencha User Group





<https://www.facebook.com/jungjuwon>



jjwcom@nate.com

<https://github.com/jeongjuwon/planche>



 jeongjuwon / **planche**

MySQL GUI client tool based on ExtJS

 78 commits  1 branch  0 releases  1 contributor

 branch: master ▼ **planche** / + 

History, Info 탭 스크롤링 수정

 jeongjuwon authored 3 days ago latest commit 5067cebtee 











 codemirror	코드미러 버전 업데이트	a month ago
 controller	테이블 생성 오류 수정	3 days ago
 engine	alter method 추가	6 days ago
 extjs	첫번째 미완성본 전체 소스 커	a month ago
 images	sql icon 추가	9 days ago
 lib	LIMIT 파싱 오류 수정	3 days ago
 LICENSE	Initial commit	2 months ago
 README.md	no message	9 days ago
 app.js	History, Info 탭 스크롤링 수정	3 days ago
 config-host.js	Example Host 설정 추가	a month ago



Planche for MySQL

Query

```
1 SELECT
2   'group_id','group_name','description','last_work_time','last_work_user_id'
3 FROM 'whiteSQL'.tbl_agentgroup_info;
```

Alter Table 'tbl_blocksql_list' in 'whiteSQL'

Field Name	Datatype	Length	Default	PK	Not Null	Unsigned	Auto Incr	Zerofill	Comment
agent_id	int	10							서버아이디 (=tbl_agent_info.u...
policy_name	varchar	255							정책이름
policy_type	enum	'1','2','3...							정책종류(1:BLOCK_SQL, 2:US...
policy_id	int	10							정책아이디
apply_starttime	int	10							정책적용시작시간
apply_endtime	int	10							정책적용종료시간
event_level	tinyint	4							이벤트레벨(미리사용하지 않...
description	varchar	255							설명
reg_time	int	10							정책등록시간
reg_user_id	int	10							
state	char	1	A						
on_off	enum	'0','1'	0						
sync_flag	enum	'0','1'	0						
sync_succ_flag	enum	'0','1'	0						
sync_time	int	10							
sync_user_id	int	10							
last_work_time	int	10	0						
last_work_user_id	varchar	20							

웹기반의 MySQL GUI Client Tool

Query

```
8 SELECT
9   'eventlog_id','event_type','event_time','event_level','event_kind','policy_type','event_msg','event_confirm_flag','event_confirm_comment','event_tb
10 FROM 'whiteSQL'.tbl_eventlog;
```

Query

```
15 SELECT
16   'policy_id','policy_name','policy_type','policy_properties','agent_id','alarm_level','block','reg_time','reg_user_id','state','on_off','sync_flag',
17 FROM 'whiteSQL'.tbl_policy_list;
```

Paste SQL Statement
Copy Table(s) To Different Host/Database
Open Table
Create Table
Alter Table
More Table Operations
Create Trigger

Messages

Table

100 Refresh

properties	agent_id	alarm_level	block	reg_time	reg_user_id	state	on_o
MYISAM		0	1397792249	1	N	1	1
MRG_MYISAM		0	1395296817	1	A	0	1
CSV		1	1395297405	1	A	0	0
BLACKHOLE		0	1395298995	1	A	0	0
MEMORY		1	1395299032	1	A	0	0
FEDERATED		0	1395301382	1	A	0	0
ARCHIVE		0	1395303527	1	A	0	0
INNODB		0	1398073470	1	N	1	1

PERFORMANCE_SCHEMA

Total 15 Rows

FRAMEWORK



Sencha Ext JS

Windows

Window Variations
A collection of Windows in different configurations, showing headers attached to any side of the window.

Layout Window
A window containing a basic BorderLayout with nested TabPanel.

MessageBox
Different styles include confirm, alert, prompt, progress and wait and also...

Trees

Drag and Drop Reordering
A TreePanel loaded asynchronously via a JSON TreeLoader that shows drag and drop with container scroll.

Locking TreeGrid (New)
TreeGrid with lockable columns

Custom Drop Lo...
Apply custom determine who can be moved

Grids

Basic Array Grid
A basic read-only grid loaded from local array data that demonstrates the use of custom column renderer functions.

XML Grid
A simple read-only grid loaded from XML data.

Paging
A grid with paging, cross-domain data loading and custom-rendered expandable row bodies.

Grid Plugins
Multiple grids customized via plugins: expander rows, checkbox selection and row numbering.

Grid Data Binding (advanced)
Refactoring the basic data binding example to use a class-based application design model.

Grouped Header Grid
A basic grouping grid showing collapsible data groups that can be customized via the 'Group By' header menu option.

Combination Examples

Web Desktop
Demonstrates how one could build a desktop in the browser using Ext components including a module plugin system.

Portal Demo
A page layout using several custom extensions to provide a web portal interface.

Ext JS 3 & 4 on one page
This example demonstrates Ext JS 4's sandboxing behavior which allows you to run Ext JS 3 & 4 on the same page.

Image Viewer
DataView and TreePanel example that demonstrates dragging data items from a DataView into a TreePanel.

Theme Viewer
View and test every Ext component against bundled Ext themes.

Simple Tasks
Complete personal task management application example

Ext JS Calendar

Example Calendar application. Demonstrates the new Day, Week and Month views and how to combine them.

<http://www.sencha.com/products/extjs/>



Why Ext JS?

(완벽한)컴포넌트 기반

마크업(X)

데스크탑 APP(같은..)

<http://youtu.be/QmQzCi6MqU4>



EXAMPLES



DOCUMENTATION

RELEASE NOTES

UPGRADE OVERVIEW

COMBINATION EXAMPLES

**Kitchen Sink**

Showcase of Ext JS components and features

**Sencha Charts Kitchen Sink**

Showcase of Sencha Charts

**Executive Dashboard**

A Tablet-Friendly Responsive Dashboard Application

**Feed Viewer**

RSS feed reader example application

**Ext JS Calendar**

Example Calendar application

**Web Desktop**

Demonstrates how one could build a desktop in the browser using Ext JS

**Ticket App**

Simple ticket application that demonstrates MVC concepts and data binding

**Portal Demo**

A page layout using several custom extensions to provide a web portal interface.

**Simple Tasks**

Complete personal task management application example

**Responsive Design Demo**

A simple application that shows how to implement a responsive UI.

**Image Viewer**

An image viewer with drag/drop functionality

**Theme Viewer**

View and test every Ext component against bundled Ext themes.

**Right-to-Left (RTL)**

Demonstrates Ext JS support for right-to-left languages

**Ext JS Legacy Charts Kitchen Sink**

Showcase of Ext JS Legacy Charts

```

1 // The bindings defined specifically in the Sublime Text mode
2 var bindings = {
3   "Alt-Left": "goSubwordLeft",
4   "Alt-Right": "goSubwordRight",
5   "Ctrl-Up": "scrollLineUp",
6   "Ctrl-Down": "scrollLineDown",
7   "Shift-Ctrl-L": "splitSelectionByLine",
8   "Shift-Tab": "indentLess",
9   "Esc": "singleSelectionTop",
10  "Ctrl-L": "selectLine",
11  "Shift-Ctrl-K": "deleteLine",
12  "Ctrl-Enter": "insertLineAfter",
13  "Shift-Ctrl-Enter": "insertLineBefore",
14  "Ctrl-D": "selectNextOccurrence",
15  "Shift-Ctrl-Space": "selectScope",
16  "Shift-Ctrl-M": "selectBetweenBrackets",
17  "Ctrl-M": "goToBracket",
18  "Shift-Ctrl-Up": "swapLineUp",
19  "Shift-Ctrl-Down": "swapLineDown",
20  "Ctrl-Tab": "switchToNextTab",
21  "Ctrl-Shift-Tab": "switchToPreviousTab",
22  "Shift": "selectAll",
23  "Ctrl-Alt-F": "find",
24  "F9": "toggleFullScreen",
25  "Ctrl-Alt-I": "toggleInspector",
26  "F2": "toggleCompareView",
27  "Shift-Alt-F": "findAndReplace",
28  "Ctrl-Alt-R": "replaceAll",
29  "Shift-Alt-F": "findAndReplace",
30  "Alt-F": "findAndReplace",
31

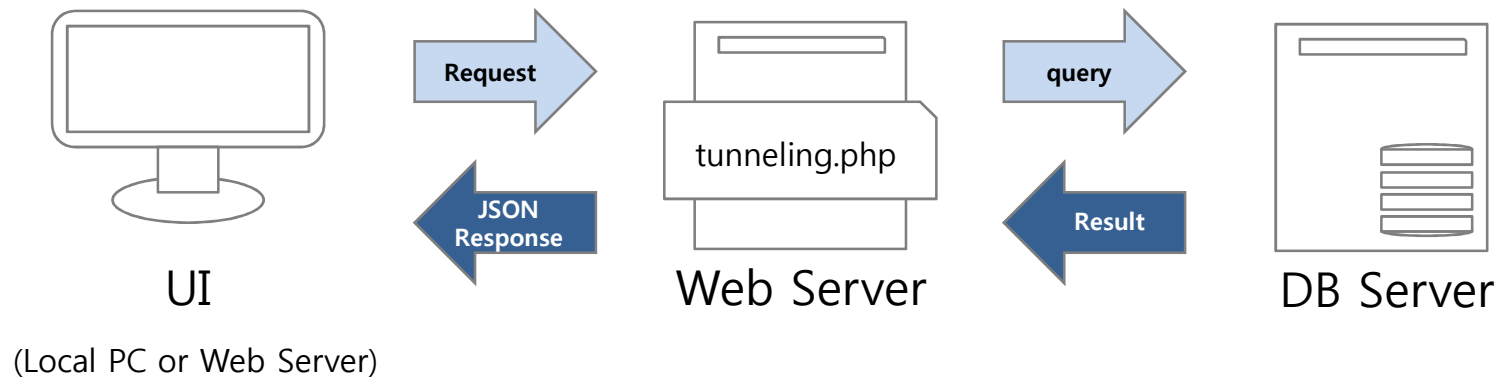
```

- Javascript기반의 Text Editor
- **Planche에서 Query Editor로 쓰임**
- 60개 이상의 다양한 언어 편집 지원
- 다양한 API, PLUGIN 지원

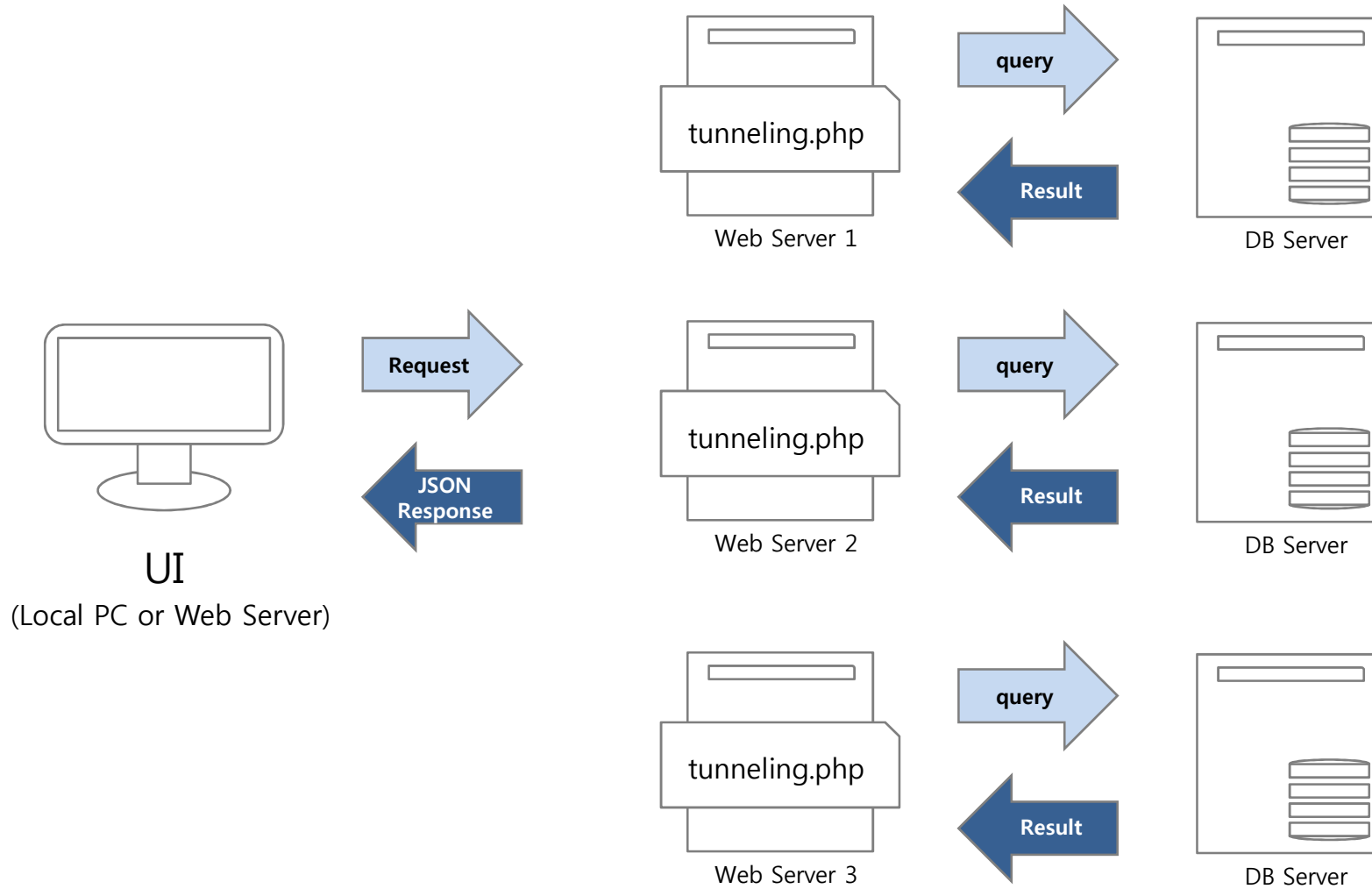


CONCEPT

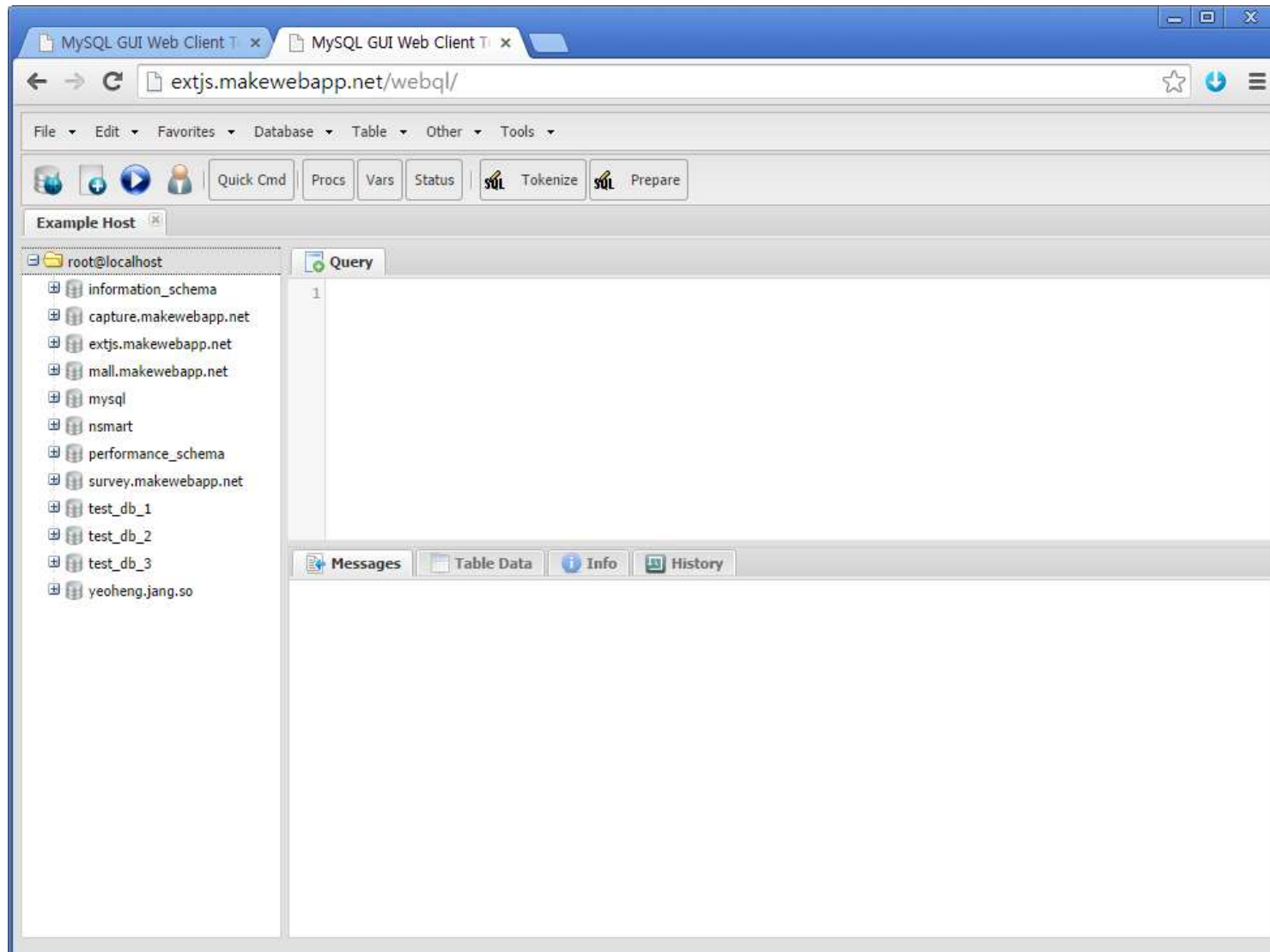
- HTTP 터널링



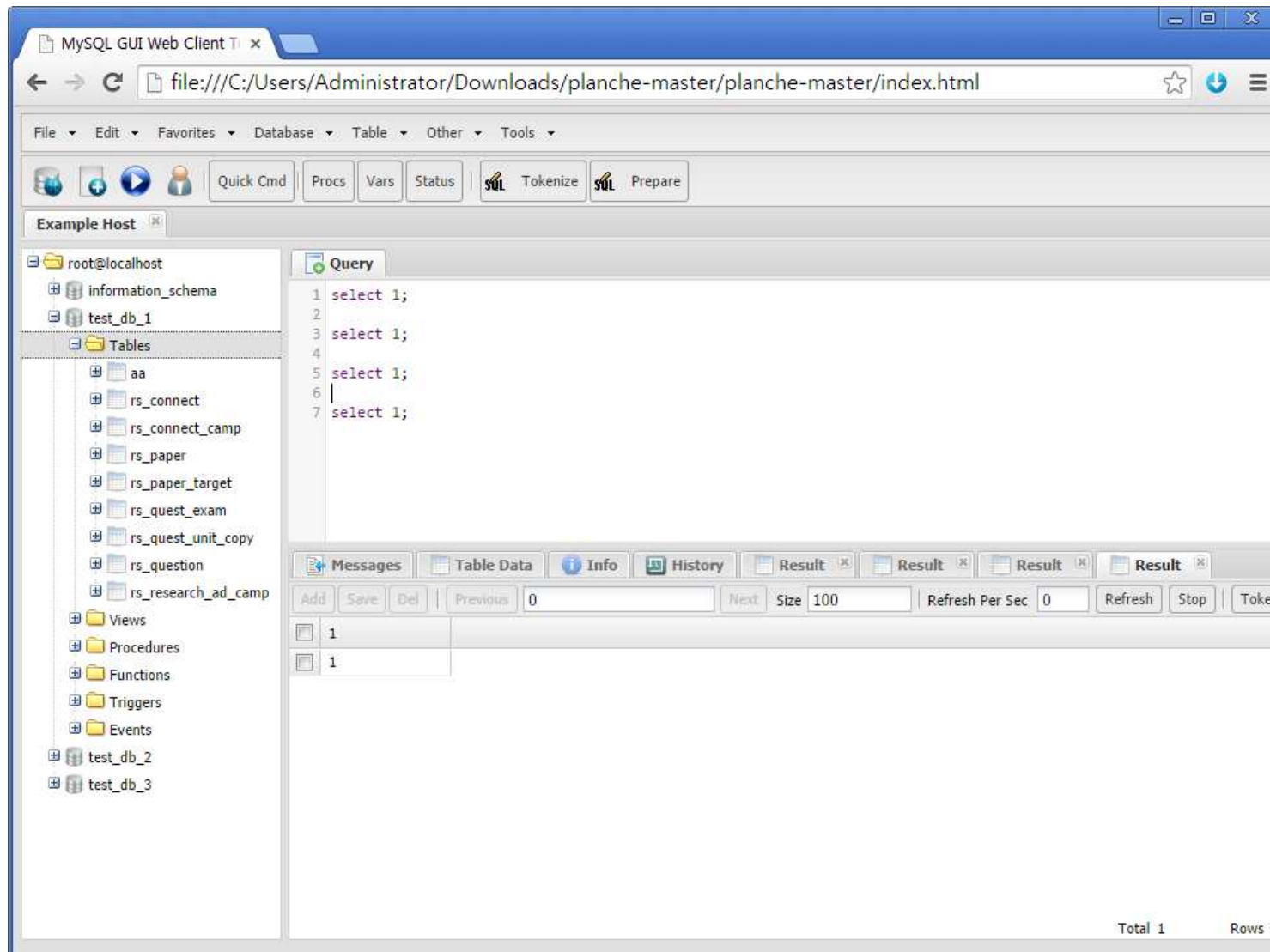
- HTTP 터널링



- 기존 처럼 웹서버에 설치 후 URL 입력



- PC에 다운 받은 상태에서 바로 실행



- AJAX를 사용하지 않음



Cross origin requests are only **supported for HTTP**.

- JSONP 사용



```

1 Ext.namespace('Planche');
2
3 Ext.application({
4     appFolder : '.',
5     name      : 'Planche',
6     history   : [],
7     launch    : function() {
8
9         //# ./lib/Window.js
10        //# 편리하지만 내부적으로 AJAX 호출
11        Ext.create('Planche.lib.Window', {
12            width : 800,
13            height : 500
14        });
15
16        //# AJAX 호출
17        Ext.Ajax.request({
18            url: 'tunneling.php',
19            params: {
20                host : 'host',
21                user  : 'user',
22                pass  : 'pass',
23                db    : 'db',
24                query  : 'query'
25            },
26            success: function(response){
27
28                //todo
29            }
30        });
31    }
32 });
33
34

```

JSONP

```

1 Ext.namespace('Planche');
2
3 Ext.application({
4     appFolder : '.',
5     name      : 'Planche',
6     history   : [],
7     launch    : function() {
8
9         //# ./lib/Window.js
10        //# Ext.require 이용
11        Ext.require('Planche.lib.Window', function(){
12
13            Ext.create('Planche.lib.Window', {
14                width : 800,
15                height : 500
16            });
17
18            }, this);
19
20        //# JSONP 호출
21        Ext.data.JsonP.request({
22            url: 'tunneling.php',
23            params: {
24                host : 'host',
25                user  : 'user',
26                pass  : 'pass',
27                db    : 'db',
28                query  : 'query'
29            },
30            callbackKey : 'callback',
31            success: function(response) {
32
33                //todo
34            }
35        });
36    }
37 });

```

CONNECTION



Connection

호스트 설정

Connect to MySQL Host

Host Name	Host	User	Charset	Port	HTTP Tunneling URL
Local Example	localhost	root	utf8	3306	http://192.168.0.20/tunnel.php
Example Host	localhost	planche	utf8	3306	http://extjs.makewebapp.net/webql/planche_tnl.php

```
1 var Planche = Planche || {};  
2 Planche.config = {  
3   hosts : [  
4     {  
5       host_name : 'Local Example',  
6       http_tunneling : 'http://192.168.0.20/tunnel.php',  
7       host : 'localhost',  
8       user : 'root',  
9       pass : 'password',  
10      charset : 'utf8',  
11      port : 3306  
12    },  
13    {  
14      host_name : 'Example Host',  
15      http_tunneling : 'http://extjs.makewebapp.net/webql/planche_tnl.php',  
16      host : 'localhost',  
17      user : 'planche',  
18      pass : 'planche',  
19      charset : 'utf8',  
20      port : 3306  
21    }  
22  ]  
23 }  
24  
25
```

LAYOUT



레이아웃

메뉴

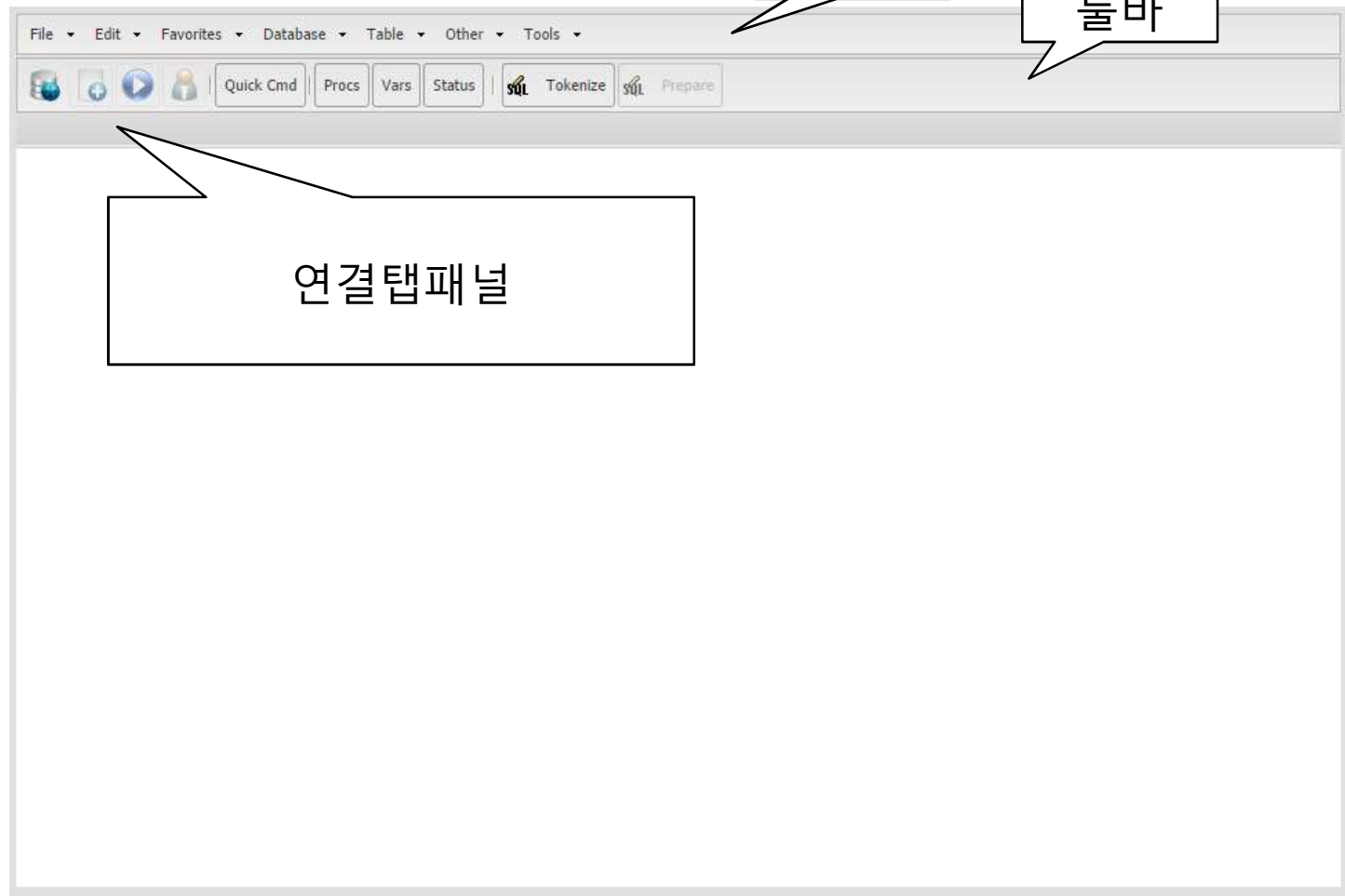
툴바

연결탭패널

메뉴

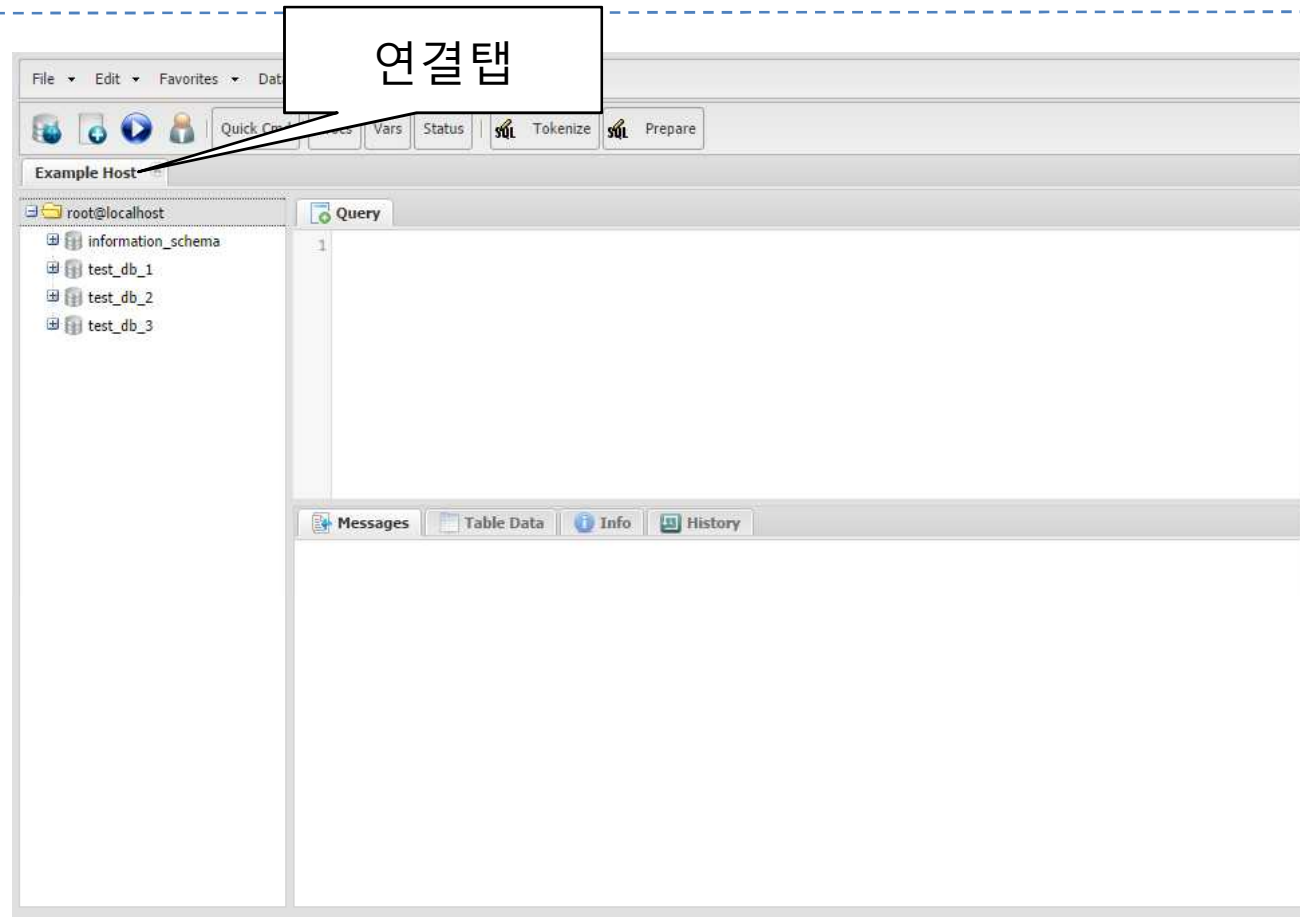
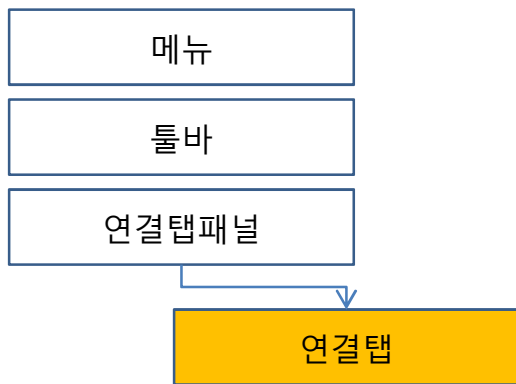
툴바

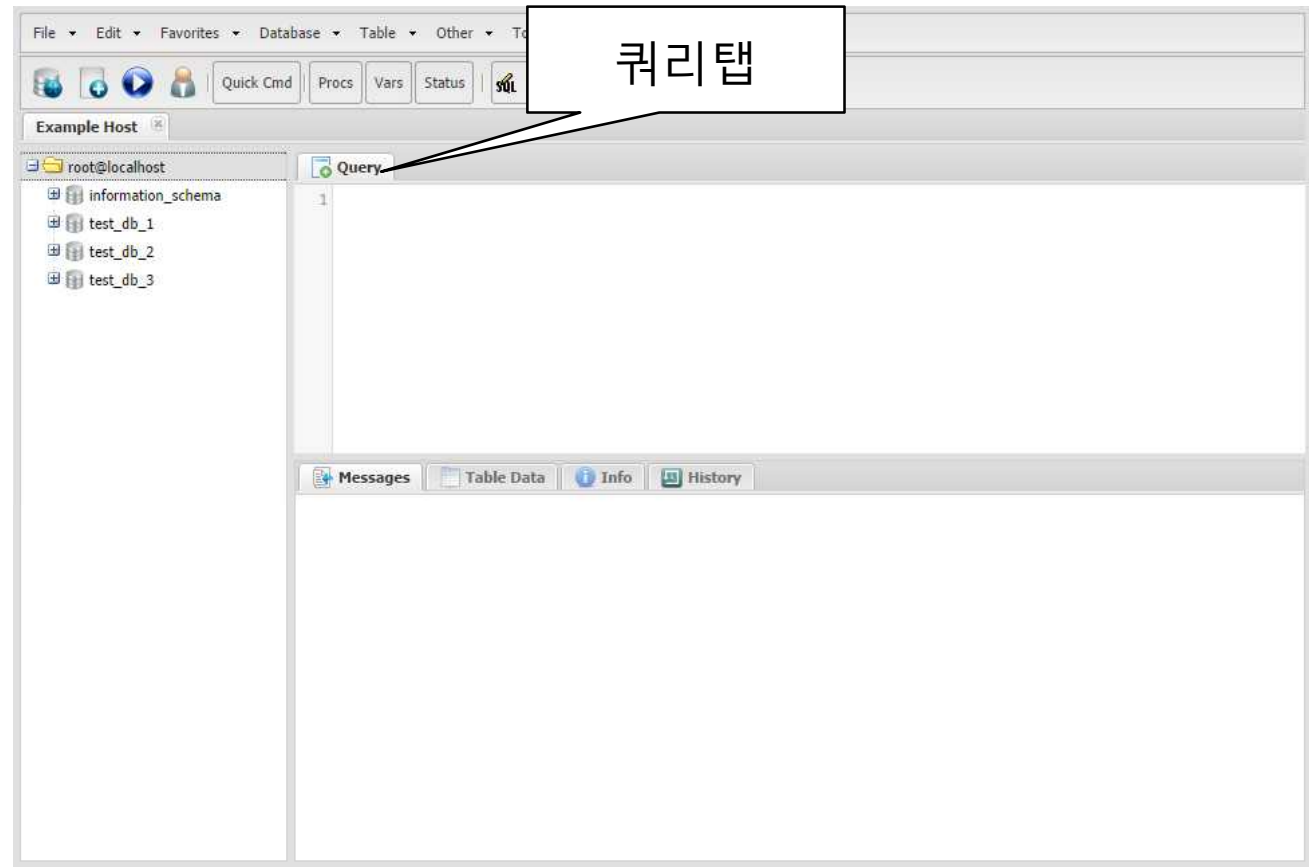
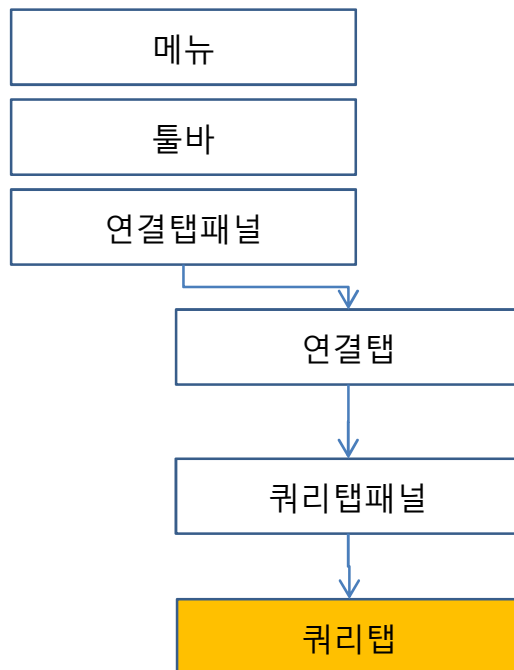
연결탭패널

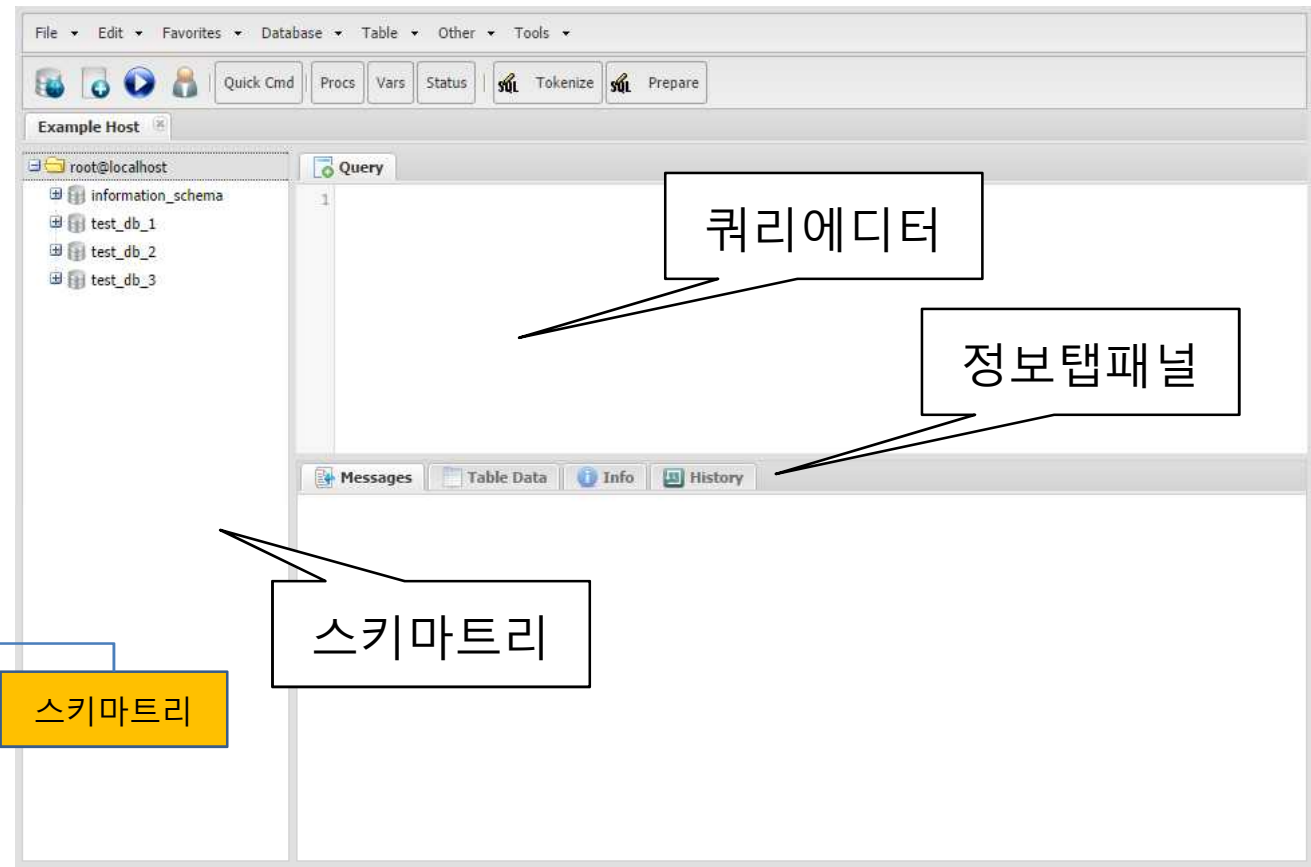
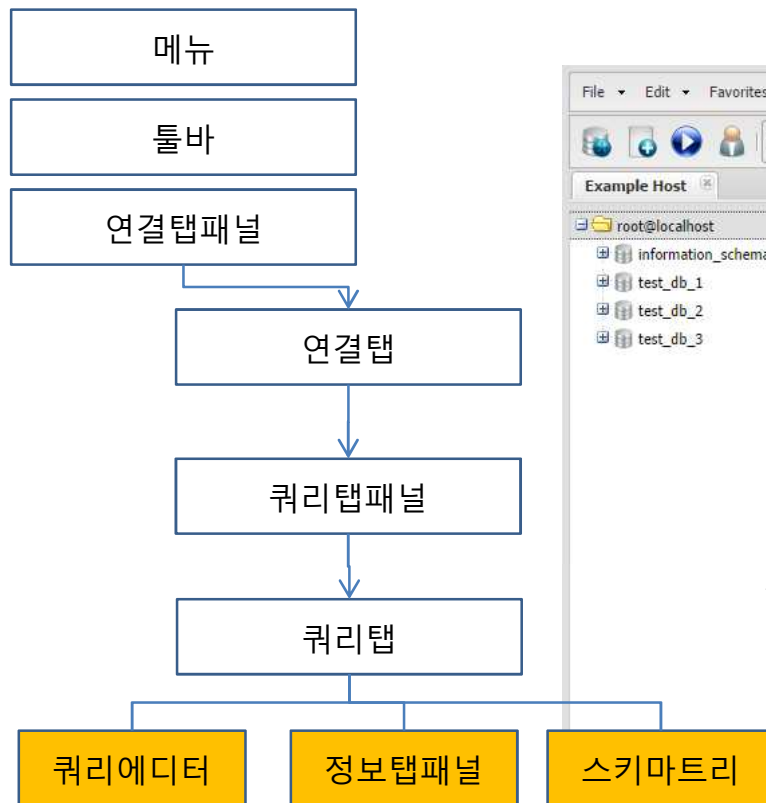


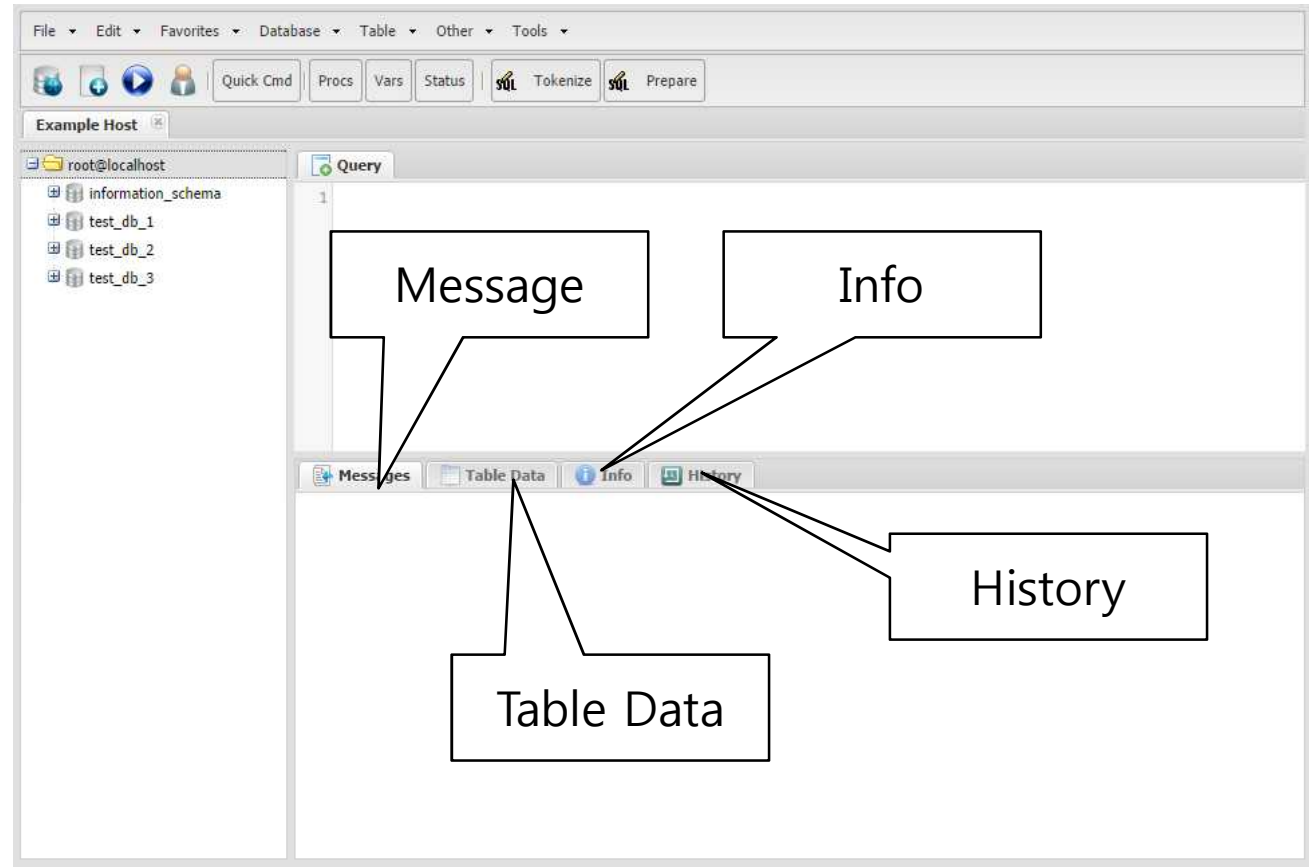
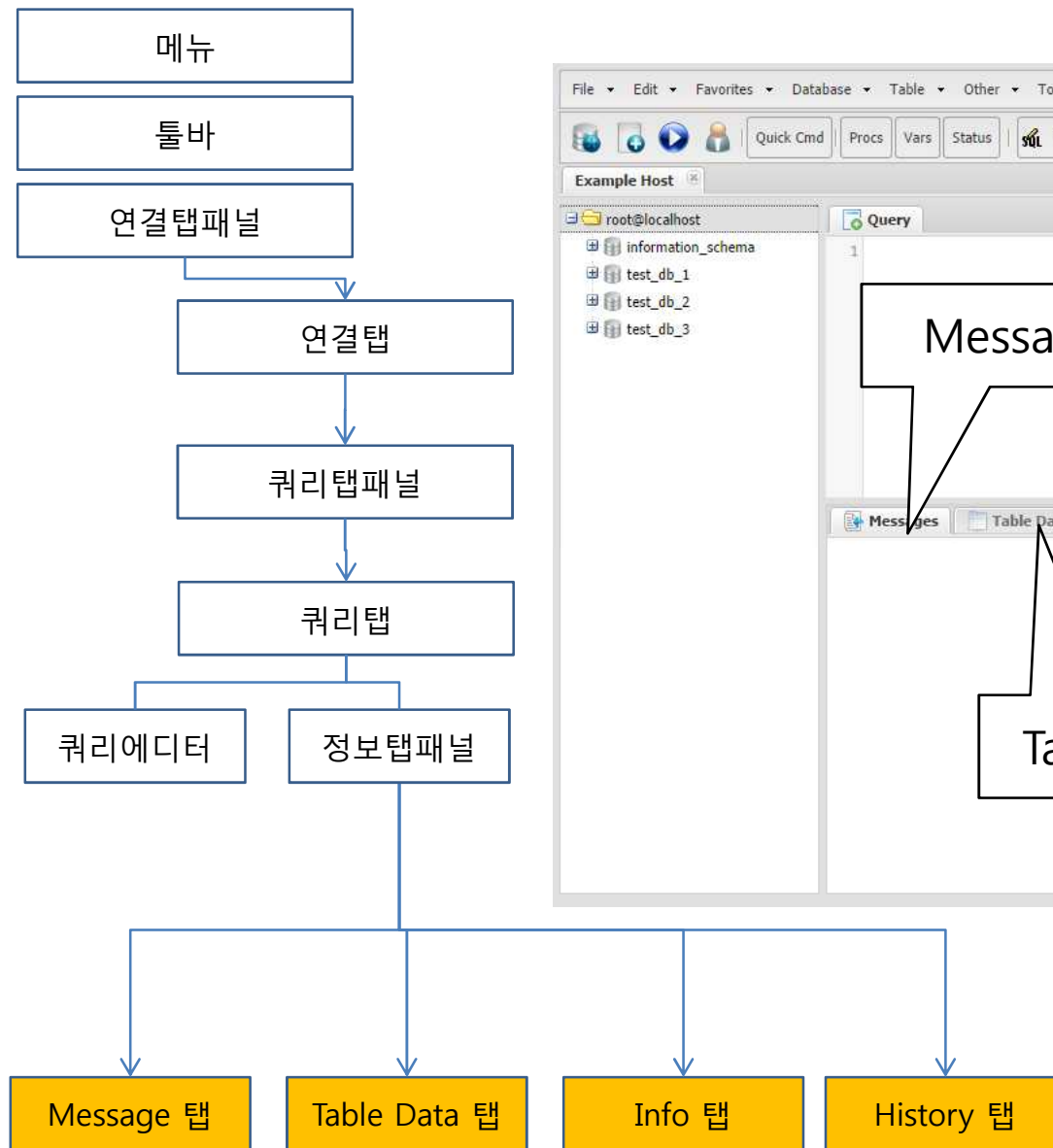


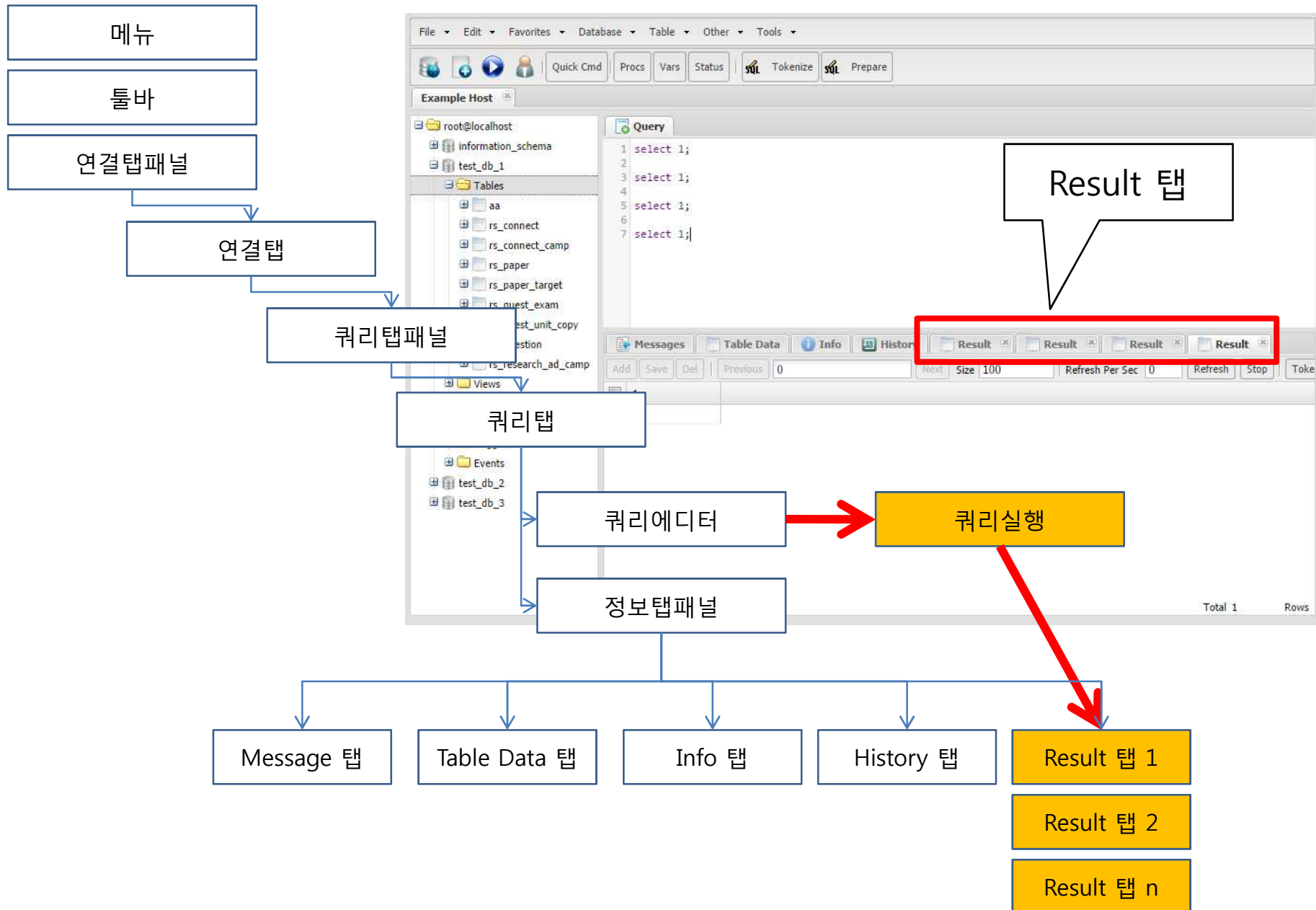
레이아웃







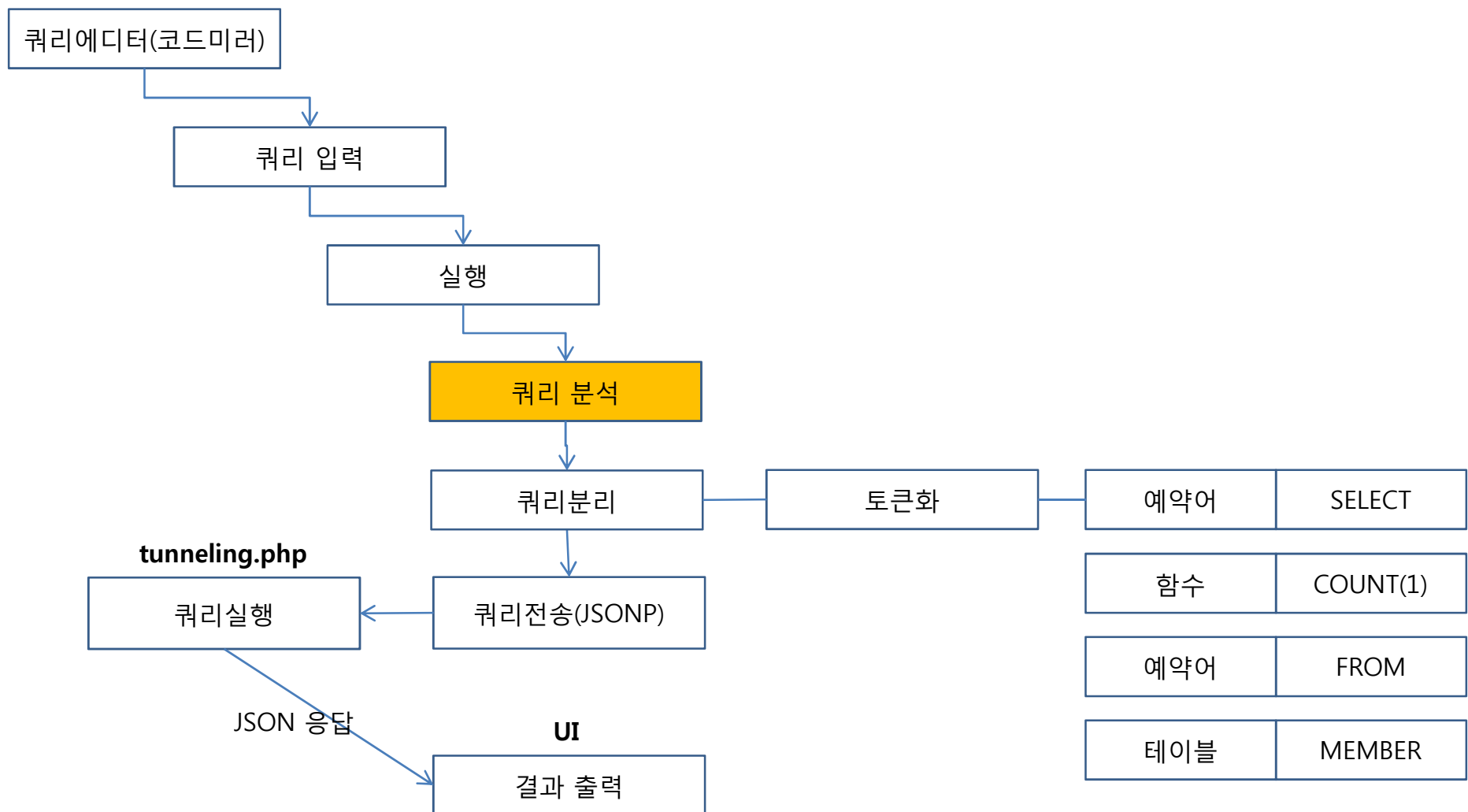




쿼리 실행



쿼리 실행 구조





UI에 연결된 쿼리 처리 방법

Ext.String.format을 이용한 쿼리 치환

```
1 Ext.define('Planche.engine.MySQL', function(){
2
3   var queries = {
4     SELECT_USER : 'SELECT * FROM `mysql`.`user`',
5     SHOW_FULL_FIELDS : 'SHOW FULL FIELDS FROM `{0}`.`{1}`',
6     SHOW_ADVANCED_PROPERTIES : 'SHOW TABLE STATUS FROM `{0}` LIKE "{1}"',
7     SHOW_DATABASES : 'SHOW DATABASES',
8     OPEN_TABLE : 'SELECT * FROM `{0}`.`{1}`',
9     SHOW_PROCEDURES : 'SHOW PROCEDURE STATUS WHERE DB = "{0}"',
10    SHOW_FUNCTIONS : 'SHOW FUNCTION STATUS WHERE DB = "{0}"',
11    SHOW_TRIGGERS : 'SHOW TRIGGERS FROM `{0}`',
12    SHOW_VARIABLES : 'SHOW VARIABLES',
13    SHOW_STATUS : 'SHOW STATUS',
14    SHOW_GLOBAL_STATUS : 'SHOW GLOBAL STATUS',
15    SHOW_SESSION_STATUS : 'SHOW SESSION STATUS'
16  };
17
18  var joins = ['NATURAL JOIN', 'INNER JOIN', 'LEFT INNER JOIN', 'RIGHT INNER JOIN', 'LEFT
19  var functions = ['ABS', 'ACOS', 'ADDDATE', 'ADDTIME', 'AES_DECRYPT', 'AES_ENCRYPT', 'AREA',
20  var reserved_words = ['ACCESSIBLE', 'ADD', 'ALL', 'ALTER', 'ANALYZE', 'AND', 'AS', 'ASC', 'ASENS
21  var data_types = ['tinyint', 'int', 'varchar', 'float', 'double', 'timestamp', 'bit', 'bigint
22
23  var regexLimit = "LIMIT\\s+[0-9]+((\\s+?,|,)?(\\s+)?[0-9]+)";
24
25  return {
26    singleton: true,
27    constructor: function(config) {
28      this.callParent(arguments);
29    },
30    getQuery : function(query){
31      var args = Ext.Array.slice(arguments, 1);
32      args.unshift(queries[query]);
33      return Ext.String.format.apply(this, args);
34    },
35    getDataTypes : function(){
36      return data_types;
37    },
38    getDataTypesToJSON : function(){
39      var json = [];
40      Ext.Array.each(data_types, function(type, idx){
41        json.push([type, type]);
42      });
43      return json;
44    }
45  };
46
47  return {
48    singleton: true,
49    constructor: function(config) {
50      this.callParent(arguments);
51    },
52    getQuery : function(query){
53      var args = Ext.Array.slice(arguments, 1);
54      args.unshift(queries[query]);
55      return Ext.String.format.apply(this, args);
56    },
57    getDataTypes : function(){
58      return data_types;
59    },
60    getDataTypesToJSON : function(){
61      var json = [];
62      Ext.Array.each(data_types, function(type, idx){
63        json.push([type, type]);
64      });
65      return json;
66    }
67  };
68
69  return {
70    singleton: true,
71    constructor: function(config) {
72      this.callParent(arguments);
73    },
74    getQuery : function(query){
75      var args = Ext.Array.slice(arguments, 1);
76      args.unshift(queries[query]);
77      return Ext.String.format.apply(this, args);
78    },
79    getDataTypes : function(){
80      return data_types;
81    },
82    getDataTypesToJSON : function(){
83      var json = [];
84      Ext.Array.each(data_types, function(type, idx){
85        json.push([type, type]);
86      });
87      return json;
88    }
89  };
90
91  return {
92    singleton: true,
93    constructor: function(config) {
94      this.callParent(arguments);
95    },
96    getQuery : function(query){
97      var args = Ext.Array.slice(arguments, 1);
98      args.unshift(queries[query]);
99      return Ext.String.format.apply(this, args);
100    },
101    getDataTypes : function(){
102      return data_types;
103    },
104    getDataTypesToJSON : function(){
105      var json = [];
106      Ext.Array.each(data_types, function(type, idx){
107        json.push([type, type]);
108      });
109      return json;
110    }
111  };
112
113  return {
114    singleton: true,
115    constructor: function(config) {
116      this.callParent(arguments);
117    },
118    getQuery : function(query){
119      var args = Ext.Array.slice(arguments, 1);
120      args.unshift(queries[query]);
121      return Ext.String.format.apply(this, args);
122    },
123    getDataTypes : function(){
124      return data_types;
125    },
126    getDataTypesToJSON : function(){
127      var json = [];
128      Ext.Array.each(data_types, function(type, idx){
129        json.push([type, type]);
130      });
131      return json;
132    }
133  };
134
135  return {
136    singleton: true,
137    constructor: function(config) {
138      this.callParent(arguments);
139    },
140    getQuery : function(query){
141      var args = Ext.Array.slice(arguments, 1);
142      args.unshift(queries[query]);
143      return Ext.String.format.apply(this, args);
144    },
145    getDataTypes : function(){
146      return data_types;
147    },
148    getDataTypesToJSON : function(){
149      var json = [];
150      Ext.Array.each(data_types, function(type, idx){
151        json.push([type, type]);
152      });
153      return json;
154    }
155  };
156
157  return {
158    singleton: true,
159    constructor: function(config) {
160      this.callParent(arguments);
161    },
162    getQuery : function(query){
163      var args = Ext.Array.slice(arguments, 1);
164      args.unshift(queries[query]);
165      return Ext.String.format.apply(this, args);
166    },
167    getDataTypes : function(){
168      return data_types;
169    },
170    getDataTypesToJSON : function(){
171      var json = [];
172      Ext.Array.each(data_types, function(type, idx){
173        json.push([type, type]);
174      });
175      return json;
176    }
177  };
178
179  return {
180    singleton: true,
181    constructor: function(config) {
182      this.callParent(arguments);
183    },
184    getQuery : function(query){
185      var args = Ext.Array.slice(arguments, 1);
186      args.unshift(queries[query]);
187      return Ext.String.format.apply(this, args);
188    },
189    getDataTypes : function(){
190      return data_types;
191    },
192    getDataTypesToJSON : function(){
193      var json = [];
194      Ext.Array.each(data_types, function(type, idx){
195        json.push([type, type]);
196      });
197      return json;
198    }
199  };
200
201  return {
202    singleton: true,
203    constructor: function(config) {
204      this.callParent(arguments);
205    },
206    getQuery : function(query){
207      var args = Ext.Array.slice(arguments, 1);
208      args.unshift(queries[query]);
209      return Ext.String.format.apply(this, args);
210    },
211    getDataTypes : function(){
212      return data_types;
213    },
214    getDataTypesToJSON : function(){
215      var json = [];
216      Ext.Array.each(data_types, function(type, idx){
217        json.push([type, type]);
218      });
219      return json;
220    }
221  };
222
223  return {
224    singleton: true,
225    constructor: function(config) {
226      this.callParent(arguments);
227    },
228    getQuery : function(query){
229      var args = Ext.Array.slice(arguments, 1);
230      args.unshift(queries[query]);
231      return Ext.String.format.apply(this, args);
232    },
233    getDataTypes : function(){
234      return data_types;
235    },
236    getDataTypesToJSON : function(){
237      var json = [];
238      Ext.Array.each(data_types, function(type, idx){
239        json.push([type, type]);
240      });
241      return json;
242    }
243  };
244
245  return {
246    singleton: true,
247    constructor: function(config) {
248      this.callParent(arguments);
249    },
250    getQuery : function(query){
251      var args = Ext.Array.slice(arguments, 1);
252      args.unshift(queries[query]);
253      return Ext.String.format.apply(this, args);
254    },
255    getDataTypes : function(){
256      return data_types;
257    },
258    getDataTypesToJSON : function(){
259      var json = [];
260      Ext.Array.each(data_types, function(type, idx){
261        json.push([type, type]);
262      });
263      return json;
264    }
265  };
266
267  return {
268    singleton: true,
269    constructor: function(config) {
270      this.callParent(arguments);
271    },
272    getQuery : function(query){
273      var args = Ext.Array.slice(arguments, 1);
274      args.unshift(queries[query]);
275      return Ext.String.format.apply(this, args);
276    },
277    getDataTypes : function(){
278      return data_types;
279    },
280    getDataTypesToJSON : function(){
281      var json = [];
282      Ext.Array.each(data_types, function(type, idx){
283        json.push([type, type]);
284      });
285      return json;
286    }
287  };
288
289  return {
290    singleton: true,
291    constructor: function(config) {
292      this.callParent(arguments);
293    },
294    getQuery : function(query){
295      var args = Ext.Array.slice(arguments, 1);
296      args.unshift(queries[query]);
297      return Ext.String.format.apply(this, args);
298    },
299    getDataTypes : function(){
300      return data_types;
301    },
302    getDataTypesToJSON : function(){
303      var json = [];
304      Ext.Array.each(data_types, function(type, idx){
305        json.push([type, type]);
306      });
307      return json;
308    }
309  };
310
311  return {
312    singleton: true,
313    constructor: function(config) {
314      this.callParent(arguments);
315    },
316    getQuery : function(query){
317      var args = Ext.Array.slice(arguments, 1);
318      args.unshift(queries[query]);
319      return Ext.String.format.apply(this, args);
320    },
321    getDataTypes : function(){
322      return data_types;
323    },
324    getDataTypesToJSON : function(){
325      var json = [];
326      Ext.Array.each(data_types, function(type, idx){
327        json.push([type, type]);
328      });
329      return json;
330    }
331  };
332
333  return {
334    singleton: true,
335    constructor: function(config) {
336      this.callParent(arguments);
337    },
338    getQuery : function(query){
339      var args = Ext.Array.slice(arguments, 1);
340      args.unshift(queries[query]);
341      return Ext.String.format.apply(this, args);
342    },
343    getDataTypes : function(){
344      return data_types;
345    },
346    getDataTypesToJSON : function(){
347      var json = [];
348      Ext.Array.each(data_types, function(type, idx){
349        json.push([type, type]);
350      });
351      return json;
352    }
353  };
354
355  return {
356    singleton: true,
357    constructor: function(config) {
358      this.callParent(arguments);
359    },
360    getQuery : function(query){
361      var args = Ext.Array.slice(arguments, 1);
362      args.unshift(queries[query]);
363      return Ext.String.format.apply(this, args);
364    },
365    getDataTypes : function(){
366      return data_types;
367    },
368    getDataTypesToJSON : function(){
369      var json = [];
370      Ext.Array.each(data_types, function(type, idx){
371        json.push([type, type]);
372      });
373      return json;
374    }
375  };
376
377  return {
378    singleton: true,
379    constructor: function(config) {
380      this.callParent(arguments);
381    },
382    getQuery : function(query){
383      var args = Ext.Array.slice(arguments, 1);
384      args.unshift(queries[query]);
385      return Ext.String.format.apply(this, args);
386    },
387    getDataTypes : function(){
388      return data_types;
389    },
390    getDataTypesToJSON : function(){
391      var json = [];
392      Ext.Array.each(data_types, function(type, idx){
393        json.push([type, type]);
394      });
395      return json;
396    }
397  };
398
399  return {
400    singleton: true,
401    constructor: function(config) {
402      this.callParent(arguments);
403    },
404    getQuery : function(query){
405      var args = Ext.Array.slice(arguments, 1);
406      args.unshift(queries[query]);
407      return Ext.String.format.apply(this, args);
408    },
409    getDataTypes : function(){
410      return data_types;
411    },
412    getDataTypesToJSON : function(){
413      var json = [];
414      Ext.Array.each(data_types, function(type, idx){
415        json.push([type, type]);
416      });
417      return json;
418    }
419  };
420
421  return {
422    singleton: true,
423    constructor: function(config) {
424      this.callParent(arguments);
425    },
426    getQuery : function(query){
427      var args = Ext.Array.slice(arguments, 1);
428      args.unshift(queries[query]);
429      return Ext.String.format.apply(this, args);
430    },
431    getDataTypes : function(){
432      return data_types;
433    },
434    getDataTypesToJSON : function(){
435      var json = [];
436      Ext.Array.each(data_types, function(type, idx){
437        json.push([type, type]);
438      });
439      return json;
440    }
441  };
442
443  return {
444    singleton: true,
445    constructor: function(config) {
446      this.callParent(arguments);
447    },
448    getQuery : function(query){
449      var args = Ext.Array.slice(arguments, 1);
450      args.unshift(queries[query]);
451      return Ext.String.format.apply(this, args);
452    },
453    getDataTypes : function(){
454      return data_types;
455    },
456    getDataTypesToJSON : function(){
457      var json = [];
458      Ext.Array.each(data_types, function(type, idx){
459        json.push([type, type]);
460      });
461      return json;
462    }
463  };
464
465  return {
466    singleton: true,
467    constructor: function(config) {
468      this.callParent(arguments);
469    },
470    getQuery : function(query){
471      var args = Ext.Array.slice(arguments, 1);
472      args.unshift(queries[query]);
473      return Ext.String.format.apply(this, args);
474    },
475    getDataTypes : function(){
476      return data_types;
477    },
478    getDataTypesToJSON : function(){
479      var json = [];
480      Ext.Array.each(data_types, function(type, idx){
481        json.push([type, type]);
482      });
483      return json;
484    }
485  };
486
487  return {
488    singleton: true,
489    constructor: function(config) {
490      this.callParent(arguments);
491    },
492    getQuery : function(query){
493      var args = Ext.Array.slice(arguments, 1);
494      args.unshift(queries[query]);
495      return Ext.String.format.apply(this, args);
496    },
497    getDataTypes : function(){
498      return data_types;
499    },
500    getDataTypesToJSON : function(){
501      var json = [];
502      Ext.Array.each(data_types, function(type, idx){
503        json.push([type, type]);
504      });
505      return json;
506    }
507  };
508
509  return {
510    singleton: true,
511    constructor: function(config) {
512      this.callParent(arguments);
513    },
514    getQuery : function(query){
515      var args = Ext.Array.slice(arguments, 1);
516      args.unshift(queries[query]);
517      return Ext.String.format.apply(this, args);
518    },
519    getDataTypes : function(){
520      return data_types;
521    },
522    getDataTypesToJSON : function(){
523      var json = [];
524      Ext.Array.each(data_types, function(type, idx){
525        json.push([type, type]);
526      });
527      return json;
528    }
529  };
530
531  return {
532    singleton: true,
533    constructor: function(config) {
534      this.callParent(arguments);
535    },
536    getQuery : function(query){
537      var args = Ext.Array.slice(arguments, 1);
538      args.unshift(queries[query]);
539      return Ext.String.format.apply(this, args);
540    },
541    getDataTypes : function(){
542      return data_types;
543    },
544    getDataTypesToJSON : function(){
545      var json = [];
546      Ext.Array.each(data_types, function(type, idx){
547        json.push([type, type]);
548      });
549      return json;
550    }
551  };
552
553  return {
554    singleton: true,
555    constructor: function(config) {
556      this.callParent(arguments);
557    },
558    getQuery : function(query){
559      var args = Ext.Array.slice(arguments, 1);
560      args.unshift(queries[query]);
561      return Ext.String.format.apply(this, args);
562    },
563    getDataTypes : function(){
564      return data_types;
565    },
566    getDataTypesToJSON : function(){
567      var json = [];
568      Ext.Array.each(data_types, function(type, idx){
569        json.push([type, type]);
570      });
571      return json;
572    }
573  };
574
575  return {
576    singleton: true,
577    constructor: function(config) {
578      this.callParent(arguments);
579    },
580    getQuery : function(query){
581      var args = Ext.Array.slice(arguments, 1);
582      args.unshift(queries[query]);
583      return Ext.String.format.apply(this, args);
584    },
585    getDataTypes : function(){
586      return data_types;
587    },
588    getDataTypesToJSON : function(){
589      var json = [];
590      Ext.Array.each(data_types, function(type, idx){
591        json.push([type, type]);
592      });
593      return json;
594    }
595  };
596
597  return {
598    singleton: true,
599    constructor: function(config) {
600      this.callParent(arguments);
601    },
602    getQuery : function(query){
603      var args = Ext.Array.slice(arguments, 1);
604      args.unshift(queries[query]);
605      return Ext.String.format.apply(this, args);
606    },
607    getDataTypes : function(){
608      return data_types;
609    },
610    getDataTypesToJSON : function(){
611      var json = [];
612      Ext.Array.each(data_types, function(type, idx){
613        json.push([type, type]);
614      });
615      return json;
616    }
617  };
618
619  return {
620    singleton: true,
621    constructor: function(config) {
622      this.callParent(arguments);
623    },
624    getQuery : function(query){
625      var args = Ext.Array.slice(arguments, 1);
626      args.unshift(queries[query]);
627      return Ext.String.format.apply(this, args);
628    },
629    getDataTypes : function(){
630      return data_types;
631    },
632    getDataTypesToJSON : function(){
633      var json = [];
634      Ext.Array.each(data_types, function(type, idx){
635        json.push([type, type]);
636      });
637      return json;
638    }
639  };
640
641  return {
642    singleton: true,
643    constructor: function(config) {
644      this.callParent(arguments);
645    },
646    getQuery : function(query){
647      var args = Ext.Array.slice(arguments, 1);
648      args.unshift(queries[query]);
649      return Ext.String.format.apply(this, args);
650    },
651    getDataTypes : function(){
652      return data_types;
653    },
654    getDataTypesToJSON : function(){
655      var json = [];
656      Ext.Array.each(data_types, function(type, idx){
657        json.push([type, type]);
658      });
659      return json;
660    }
661  };
662
663  return {
664    singleton: true,
665    constructor: function(config) {
666      this.callParent(arguments);
667    },
668    getQuery : function(query){
669      var args = Ext.Array.slice(arguments, 1);
670      args.unshift(queries[query]);
671      return Ext.String.format.apply(this, args);
672    },
673    getDataTypes : function(){
674      return data_types;
675    },
676    getDataTypesToJSON : function(){
677      var json = [];
678      Ext.Array.each(data_types, function(type, idx){
679        json.push([type, type]);
680      });
681      return json;
682    }
683  };
684
685  return {
686    singleton: true,
687    constructor: function(config) {
688      this.callParent(arguments);
689    },
690    getQuery : function(query){
691      var args = Ext.Array.slice(arguments, 1);
692      args.unshift(queries[query]);
693      return Ext.String.format.apply(this, args);
694    },
695    getDataTypes : function(){
696      return data_types;
697    },
698    getDataTypesToJSON : function(){
699      var json = [];
700      Ext.Array.each(data_types, function(type, idx){
701        json.push([type, type]);
702      });
703      return json;
704    }
705  };
706
707  return {
708    singleton: true,
709    constructor: function(config) {
710      this.callParent(arguments);
711    },
712    getQuery : function(query){
713      var args = Ext.Array.slice(arguments, 1);
714      args.unshift(queries[query]);
715      return Ext.String.format.apply(this, args);
716    },
717    getDataTypes : function(){
718      return data_types;
719    },
720    getDataTypesToJSON : function(){
721      var json = [];
722      Ext.Array.each(data_types, function(type, idx){
723        json.push([type, type]);
724      });
725      return json;
726    }
727  };
728
729  return {
730    singleton: true,
731    constructor: function(config) {
732      this.callParent(arguments);
733    },
734    getQuery : function(query){
735      var args = Ext.Array.slice(arguments, 1);
736      args.unshift(queries[query]);
737      return Ext.String.format.apply(this, args);
738    },
739    getDataTypes : function(){
740      return data_types;
741    },
742    getDataTypesToJSON : function(){
743      var json = [];
744      Ext.Array.each(data_types, function(type, idx){
745        json.push([type, type]);
746      });
747      return json;
748    }
749  };
750
751  return {
752    singleton: true,
753    constructor: function(config) {
754      this.callParent(arguments);
755    },
756    getQuery : function(query){
757      var args = Ext.Array.slice(arguments, 1);
758      args.unshift(queries[query]);
759      return Ext.String.format.apply(this, args);
760    },
761    getDataTypes : function(){
762      return data_types;
763    },
764    getDataTypesToJSON : function(){
765      var json = [];
766      Ext.Array.each(data_types, function(type, idx){
767        json.push([type, type]);
768      });
769      return json;
770    }
771  };
772
773  return {
774    singleton: true,
775    constructor: function(config) {
776      this.callParent(arguments);
777    },
778    getQuery : function(query){
779      var args = Ext.Array.slice(arguments, 1);
780      args.unshift(queries[query]);
781      return Ext.String.format.apply(this, args);
782    },
783    getDataTypes : function(){
784      return data_types;
785    },
786    getDataTypesToJSON : function(){
787      var json = [];
788      Ext.Array.each(data_types, function(type, idx){
789        json.push([type, type]);
790      });
791      return json;
792    }
793  };
794
795  return {
796    singleton: true,
797    constructor: function(config) {
798      this.callParent(arguments);
799    },
800    getQuery : function(query){
801      var args = Ext.Array.slice(arguments, 1);
802      args.unshift(queries[query]);
803      return Ext.String.format.apply(this, args);
804    },
805    getDataTypes : function(){
806      return data_types;
807    },
808    getDataTypesToJSON : function(){
809      var json = [];
810      Ext.Array.each(data_types, function(type, idx){
811        json.push([type, type]);
812      });
813      return json;
814    }
815  };
816
817  return {
818    singleton: true,
819    constructor: function(config) {
820      this.callParent(arguments);
821    },
822    getQuery : function(query){
823      var args = Ext.Array.slice(arguments, 1);
824      args.unshift(queries[query]);
825      return Ext.String.format.apply(this, args);
826    },
827    getDataTypes : function(){
828      return data_types;
829    },
830    getDataTypesToJSON : function(){
831      var json = [];
832      Ext.Array.each(data_types, function(type, idx){
833        json.push([type, type]);
834      });
835      return json;
836    }
837  };
838
839  return {
840    singleton: true,
841    constructor: function(config) {
842      this.callParent(arguments);
843    },
844    getQuery : function(query){
845      var args = Ext.Array.slice(arguments, 1);
846      args.unshift(queries[query]);
847      return Ext.String.format.apply(this, args);
848    },
849    getDataTypes : function(){
850      return data_types;
851    },
852    getDataTypesToJSON : function(){
853      var json = [];
854      Ext.Array.each(data_types, function(type, idx){
855        json.push([type, type]);
856      });
857      return json;
858    }
859  };
860
861  return {
862    singleton: true,
863    constructor: function(config) {
864      this.callParent(arguments);
865    },
866    getQuery : function(query){
867      var args = Ext.Array.slice(arguments, 1);
868      args.unshift(queries[query]);
869      return Ext.String.format.apply(this, args);
870    },
871    getDataTypes : function(){
872      return data_types;
873    },
874    getDataTypesToJSON : function(){
875      var json = [];
876      Ext.Array.each(data_types, function(type, idx){
877        json.push([type, type]);
878      });
879      return json;
880    }
881  };
882
883  return {
884    singleton: true,
885    constructor: function(config) {
886      this.callParent(arguments);
887    },
888    getQuery : function(query){
889      var args = Ext.Array.slice(arguments, 1);
890      args.unshift(queries[query]);
891      return Ext.String.format.apply(this, args);
892    },
893    getDataTypes : function(){
894      return data_types;
895    },
896    getDataTypesToJSON : function(){
897      var json = [];
898      Ext.Array.each(data_types, function(type, idx){
899        json.push([type, type]);
900      });
901      return json;
902    }
903  };
904
905  return {
906    singleton: true,
907    constructor: function(config) {
908      this.callParent(arguments);
909    },
910    getQuery : function(query){
911      var args = Ext.Array.slice(arguments, 1);
912      args.unshift(queries[query]);
913      return Ext.String.format.apply(this, args);
914    },
915    getDataTypes : function(){
916      return data_types;
917    },
918    getDataTypesToJSON : function(){
919      var json = [];
920      Ext.Array.each(data_types, function(type, idx){
921        json.push([type, type]);
922      });
923      return json;
924    }
925  };
926
927  return {
928    singleton: true,
929    constructor: function(config) {
930      this.callParent(arguments);
931    },
932    getQuery : function(query){
933      var args = Ext.Array.slice(arguments, 1);
934      args.unshift(queries[query]);
935      return Ext.String.format.apply(this, args);
936    },
937    getDataTypes : function(){
938      return data_types;
939    },
940    getDataTypesToJSON : function(){
941      var json = [];
942      Ext.Array.each(data_types, function(type, idx){
943        json.push([type, type]);
944      });
945      return json;
946    }
947  };
948
949  return {
950    singleton: true,
951    constructor: function(config) {
952      this.callParent(arguments);
953    },
954    getQuery : function(query){
955      var args = Ext.Array.slice(arguments, 1);
956      args.unshift(queries[query]);
957      return Ext.String.format.apply(this, args);
958    },
959    getDataTypes : function(){
960      return data_types;
961    },
962    getDataTypesToJSON : function(){
963      var json = [];
964      Ext.Array.each(data_types, function(type, idx){
965        json.push([type, type]);
966      });
967      return json;
968    }
969  };
970
971  return {
972    singleton: true,
973    constructor: function(config) {
974      this.callParent(arguments);
975    },
976    getQuery : function(query){
977      var args = Ext.Array.slice(arguments, 1);
978      args.unshift(queries[query]);
979      return Ext.String.format.apply(this, args);
980    },
981    getDataTypes : function(){
982      return data_types;
983    },
984    getDataTypesToJSON : function(){
985      var json = [];
986      Ext.Array.each(data_types, function(type, idx){
987        json.push([type, type]);
988      });
989      return json;
990    }
991  };
992
993  return {
994    singleton: true,
995    constructor: function(config) {
996      this.callParent(arguments);
997    },
998    getQuery : function(query){
999      var args = Ext.Array.slice(arguments, 1);
1000      args.unshift(queries[query]);
1001      return Ext.String.format.apply(this, args);
1002    },
1003    getDataTypes : function(){
1004      return data_types;
1005    },
1006    getDataTypesToJSON : function(){
1007      var json = [];
1008      Ext.Array.each(data_types, function(type, idx){
1009        json.push([type, type]);
1010      });
1011      return json;
1012    }
1013  };
1014
1015  return {
1016    singleton: true,
1017    constructor: function(config) {
1018      this.callParent(arguments);
1019    },
1020    getQuery : function(query){
1021      var args = Ext.Array.slice(arguments, 1);
1022      args.unshift(queries[query]);
1023      return Ext.String.format.apply(this, args);
1024    },
1025    getDataTypes : function(){
1026      return data_types;
1027    },
1028    getDataTypesToJSON : function(){
1029      var json = [];
1030      Ext.Array.each(data_types, function(type, idx){
1031        json.push([type, type]);
1032      });
1033      return json;
1034    }
1035  };
1036
1037  return {
1038    singleton: true,
1039    constructor: function(config) {
1040      this.callParent(arguments);
1041    },
1042    getQuery : function(query){
1043      var args = Ext.Array.slice(arguments, 1);
1044      args.unshift(queries[query]);
1045      return Ext.String.format.apply(this, args);
1046    },
1047    getDataTypes : function(){
1048      return data_types;
1049    },
1050    getDataTypesToJSON : function(){
1051      var json = [];
1052      Ext.Array.each(data_types, function(type, idx){
1053        json.push([type, type]);
1054      });
1055      return json;
1056    }
1057  };
1058
1059  return {
1060    singleton: true,
1061    constructor: function(config) {
1062      this.callParent(arguments);
1063    },
1064    getQuery : function(query){
1065      var args = Ext.Array.slice(arguments, 1);
1066      args.unshift(queries[query]);
1067      return Ext.String.format.apply(this, args);
1068    },
1069    getDataTypes : function(){
1070      return data_types;
1071    },
1072    getDataTypesToJSON : function(){
1073      var json = [];
1074      Ext.Array.each(data_types, function(type, idx){
1075        json.push([type, type]);
1076      });
1077      return json;
1078    }
1079  };
1080
1081  return {
1082    singleton: true,
1083    constructor: function(config) {
1084      this.callParent(arguments);
1085    },
1086    getQuery : function(query){
1087      var args = Ext.Array.slice(arguments, 1);
1088      args.unshift(queries[query]);
1089      return Ext.String.format.apply(this, args);
1090    },
1091    getDataTypes : function(){
1092      return data_types;
1093    },
1094    getDataTypesToJSON : function(){
1095      var json = [];
1096      Ext.Array.each(data_types, function(type, idx){
1097        json.push([type, type]);
1098      });
1099      return json;
1100    }
1101  };
1102
1103  return {
1104    singleton: true,
1105    constructor: function(config) {
1106      this.callParent(arguments);
1107    },
1108    getQuery : function(query){
1109      var args = Ext.Array.slice(arguments, 1);
1110      args.unshift(queries[query]);
1111      return Ext.String.format.apply(this, args);
1112    },
1113    getDataTypes : function(){
1114      return data_types;
1115    },
1116    getDataTypesToJSON : function(){
1117      var json = [];
1118      Ext.Array.each(data_types, function(type, idx){
1119        json.push([type, type]);
1120      });
1121      return json;
1122    }
1123  };
1124
1125  return {
1126    singleton: true,
1127    constructor: function(config) {
1128      this.callParent(arguments);
1129    },
1130    getQuery : function(query){
1131      var args = Ext.Array.slice(arguments, 1);
1132      args.unshift(queries[query]);
1133      return Ext.String.format.apply(this, args);
1134    },
1135    getDataTypes : function(){
1136      return data_types;
1137    },
1138    getDataTypesToJSON : function(){
1139      var json = [];
1140      Ext.Array.each(data_types, function(type, idx){
1141        json.push([type, type]);
1142      });
1143      return json;
1144    }
1145  };
1146
1147  return {
1148    singleton: true,
1149    constructor: function(config) {
1150      this.callParent(arguments);
1151    },
1152    getQuery : function(query){
1153      var args = Ext.Array.slice(arguments, 1);
1154      args.unshift(queries[query]);
1155      return Ext.String.format.apply(this, args);
1156    },
1157    getDataTypes : function(){
1158      return data_types;
1159    },
1160    getDataTypesToJSON : function(){
1161      var json = [];
1162      Ext.Array.each(data_types, function(type, idx){
1163        json.push([type, type]);
1164      });
1165      return json;
1166    }
1167  };
1168
1169  return {
1170    singleton: true,
1171    constructor: function(config) {
1172      this.callParent(arguments);
1173    },
1174    getQuery : function(query){
1175      var args = Ext.Array.slice(arguments, 1);
1176      args.unshift(queries[query]);
1177      return Ext.String.format.apply(this, args);
1178    },
1179    getDataTypes : function(){
1180      return data_types;
1181    },
1182    getDataTypesToJSON : function(){
1183      var json = [];
1184      Ext.Array.each(data_types, function(type, idx){
1185        json.push([type, type]);
1186      });
1187      return json;
1188    }
1189  };
1190
1191  return {
1192    singleton: true,
1193    constructor: function(config) {
1194      this.callParent(arguments);
1195    },
1196    getQuery : function(query){
1197      var args = Ext.Array.slice(arguments, 1);
1198      args.unshift(queries[query]);
1199      return Ext.String.format.apply(this, args);
1200    },
1201    getDataTypes : function(){
1202      return data_types;
1203    },
1204    getDataTypesToJSON : function(){
1205      var json = [];
1206      Ext.Array.each(data_types, function(type, idx){
1207        json.push([type, type]);
1208      });
1209      return json;
1210    }
1211  };
1212
1213  return {
1214    singleton: true,
1215    constructor: function(config) {
1216      this.callParent(arguments);
1217    },
1218    getQuery : function(query){
1219      var args = Ext.Array.slice(arguments, 1);
1220      args.unshift(queries[query]);
1221      return Ext.String.format.apply(this, args);
1222    },
1223    getDataTypes : function(){
1224      return data_types;
1225    },
1226    getDataTypesToJSON : function(){
1227      var json = [];
1228      Ext.Array.each(data_types, function(type, idx){
1229        json.push([type, type]);
1230      });
1231      return json;
1232    }
1233  };
1234
1235  return {
1236    singleton: true,
1237    constructor: function(config) {
1238      this.callParent(arguments);
1239    },
1240    getQuery : function(query){
1241      var args = Ext.Array.slice(arguments, 1);
1242      args.unshift(queries[query]);
1243      return Ext.String.format.apply(this, args);
1244    },
1245    getDataTypes : function(){
1246      return data_types;
1247    },
1248    getDataTypesToJSON : function(){
1249      var json = [];
1250      Ext.Array.each(data_types, function(type, idx){
1251        json.push([type, type]);
1252      });
1253      return json;
1254    }
1255  };
1256
1257  return {
1258    singleton: true,
1259    constructor: function(config) {
1260      this.callParent(arguments);
1261    },
1262    getQuery : function(query){
1263      var args = Ext.Array.slice(arguments, 1);
1264      args.unshift(queries[query]);
1265      return Ext.String.format.apply(this, args);
1266    },

```

- ▶ 커서가 위치해 있는 곳에서 바로 실행

```
Query
1 select 1;
2
3 select 2;
4
5 select 3;
6
7 select 4;
8 |
9 select 5;
10
```

- ▶ 선택되어 있는 쿼리 실행

```
Query
1 select 1;
2
3 select 2;
4
5 select 3;
6
7 select 4;
8
9 select 5;
10
```

- ▶ 다중 선택(하나는 잘못 선택) 쿼리 실행

```
Query
1 select 1;
2
3 select 2;
4
5 select 3;
6
7 select 4;
8
9 select 5;
10
```

Messages Table Data Info History Result x

ect 2

▶ You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'ect 2' at line 1

예) 프로시저 생성 쿼리 – delimiter에 의한 쿼리구분자 변경 (; => \$\$)

```
1 DELIMITER $$
2
3 USE `test_db_1`$$
4
5 DROP PROCEDURE IF EXISTS `proc3`$$
6
7 CREATE DEFINER=`websql`@`localhost` PROCEDURE `proc3`()
8 BEGIN
9
10     SELECT `uniq_id`, `camp_id`, `freq`, `uid`
11     FROM `test_db_1`.`rs_connect_camp`;
12
13 END$$
14
15 DELIMITER;
16
17 SELECT 1;
18
```


변경된 쿼리 구분자를 파악하여 제대로 완성된 쿼리만 보내야 한다.

```
1 DELIMITER $$
2
3 USE `test_db_1` $$
4
5 DROP PROCEDURE IF EXISTS `proc3` $$
6
7 CREATE DEFINER=`websql`@`localhost` PROCEDURE `proc3`()
8 BEGIN
9
10     SELECT `uniq_id`, `camp_id`, `freq`, `uid`
11     FROM `test_db_1`.`rs_connect_camp`;
12
13 END $$
14
15 DELIMITER;
16
17 SELECT 1;
18
```

쿼리배열에 저장된 쿼리를 하나씩 꺼내며 재귀한다.

```
executeQuery : function(){
    var queries = this.getParsedQuery();

    var tunneling;
    var messages = [];
    (tunneling = Ext.Function.bind(function(){
        var query = queries.shift();

        if(query) {
            this.tunneling({
                db : db,
                query : query.getSQL(),
                success : function(config, response){
                    if(response.is_result_query == true){
                        var grid = this.initQueryResult({
                            icon : 'images/icon_table.png',
                            closable : true,
                            title : 'Result'
                        }, db, query, response),
                    }
                    else {
                        var msg = response.affected_rows+' row(s) affected<br/><br/>';
                        msg += 'Execution Time : '+response.exec_time+'<br/>';
                        msg += 'Transfer Time : '+response.transfer_time+'<br/>';
                        msg += 'Total Time : '+response.total_time;
                        messages.push(query.getSQL()+'<br/><br/>'+msg);
                    }
                    this.getActiveMainTab().setLoading(false);
                    tunneling();
                },
                failure : function(config, response){
                    messages.push(query.getSQL()+'<span class=\''query_err\'>> '+response.message+'</span>');
                    this.getActiveMainTab().setLoading(false);
                    tunneling();
                }
            })
        }
    })
}
else {
    this.afterExecuteQuery(messages);
}
```



Tokenizing

쿼리 분석전 실행할 문자열(쿼리)을 토큰화하여 토큰별 정보를 구축한다.

Show Tokens	
Token Type	Token
DELIMITER	DELIMITER \$\$
SPACE	
RESERVED_WORD	USE
SPACE	
BOUNDARY	.
STRING	test_db_1
BOUNDARY	.
QUERY_END	\$\$
SPACE	
RESERVED_WORD	DROP
SPACE	
RESERVED_WORD	PROCEDURE
SPACE	
RESERVED_WORD	IF
SPACE	
RESERVED_WORD	EXISTS
SPACE	
BOUNDARY	.
STRING	proc3
BOUNDARY	.

(정렬 전)

```

1
2 SELECT `res_id`, (select 1) as type, `paper_id`, (SELECT `res_id` FROM `test_db_1`.`rs_conne
3 FROM `test_db_1`.`rs_connect` a left outer join ( select `uniq_id`, `start_time`, `end_time`

```

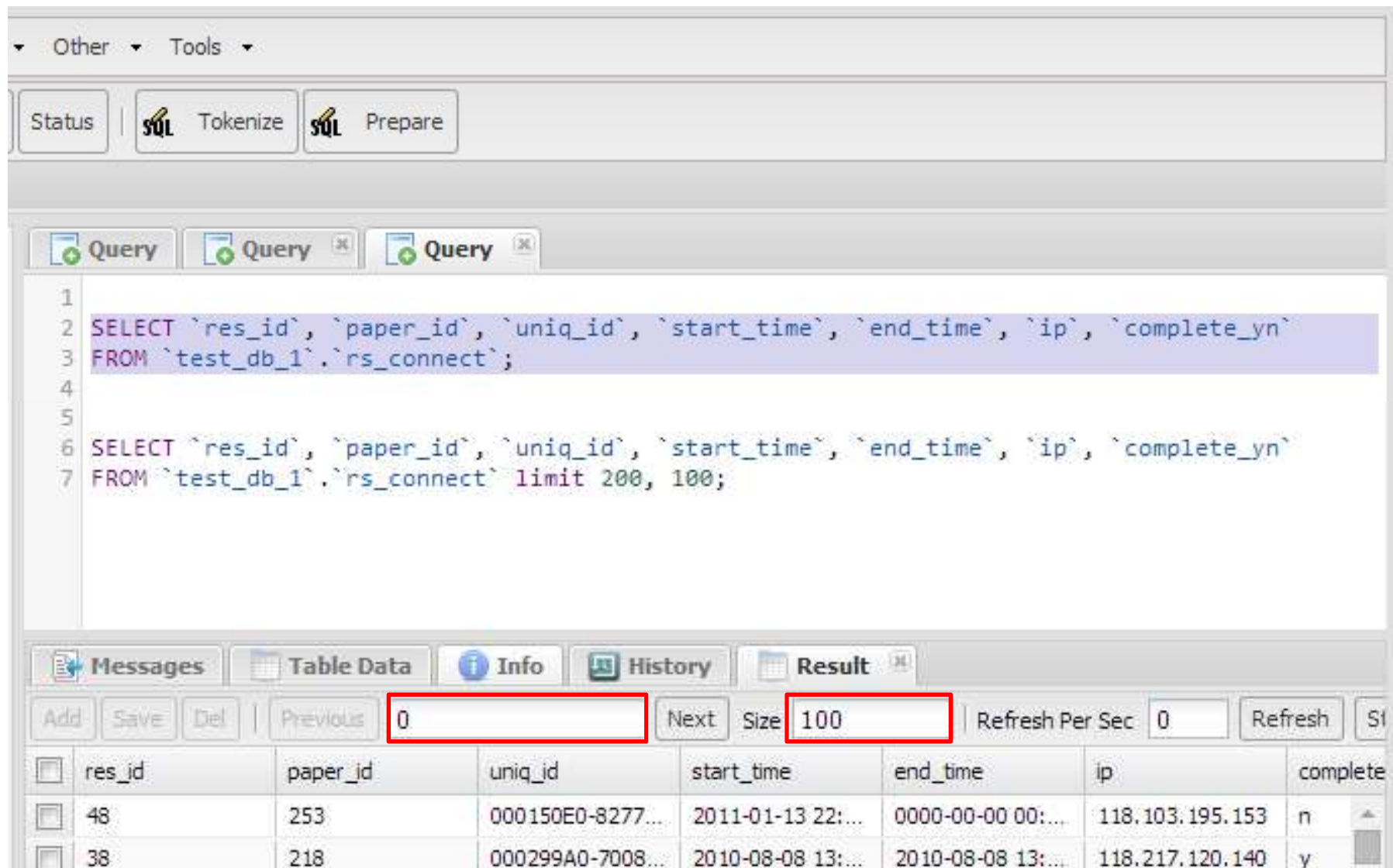
(정렬 후) 구축된 토큰별 정보로 적당한 곳에서 들여쓰기와 개행을 해준다.

```

1
2 SELECT `res_id`, (
3     SELECT 1
4 ) as type, `paper_id`, (
5     SELECT `res_id`
6     FROM `test_db_1`.`rs_connect`
7 ) as aa, `uniq_id`, `start_time`, `end_time`, `ip`, `complete_yn`
8 FROM `test_db_1`.`rs_connect` a
9 LEFT OUTER JOIN (
10
11     SELECT `uniq_id`, `start_time`, `end_time`
12     FROM `rs_connect`
13     WHERE 1=1
14 ) b on 1=1

```

LIMIT 절이 없는 경우



The screenshot shows a SQL IDE interface. The top menu bar includes 'Other' and 'Tools'. Below the menu bar are buttons for 'Status', 'SQL Tokenize', and 'SQL Prepare'. The main query editor displays two SQL queries. The first query is highlighted in blue and is a simple SELECT statement. The second query is a SELECT statement with a LIMIT clause, which is highlighted in red. Below the query editor is a toolbar with buttons for 'Messages', 'Table Data', 'Info', 'History', and 'Result'. The 'Result' tab is active, showing a table of results. The table has columns: res_id, paper_id, uniq_id, start_time, end_time, ip, and complete. The first two rows of data are visible.

```

1
2 SELECT `res_id`, `paper_id`, `uniq_id`, `start_time`, `end_time`, `ip`, `complete_yn`
3 FROM `test_db_1`.`rs_connect`;
4
5
6 SELECT `res_id`, `paper_id`, `uniq_id`, `start_time`, `end_time`, `ip`, `complete_yn`
7 FROM `test_db_1`.`rs_connect` limit 200, 100;

```

res_id	paper_id	uniq_id	start_time	end_time	ip	complete
48	253	000150E0-8277...	2011-01-13 22:...	0000-00-00 00:...	118.103.195.153	n
38	218	000299A0-7008...	2010-08-08 13:...	2010-08-08 13:...	118.217.120.140	y



쿼리 파싱 - LIMIT 절 추가

LIMIT 절이 있는 경우

OtherTools

StatusSQL TokenizeSQL Prepare

Query

Query

Query

```
1
2 SELECT `res_id`, `paper_id`, `uniq_id`, `start_time`, `end_time`, `ip`, `complete_yn`
3 FROM `test_db_1`.`rs_connect`;
4
5
6 SELECT `res_id`, `paper_id`, `uniq_id`, `start_time`, `end_time`, `ip`, `complete_yn`
7 FROM `test_db_1`.`rs_connect` limit 200, 100;
```

MessagesTable DataInfoHistoryResult

AddSaveDelPrevious200NextSize100Refresh Per Sec0RefreshSt

	res_id	paper_id	uniq_id	start_time	end_time	ip	complete
<input type="checkbox"/>	34	203	00B7EC05-A6A...	2010-06-01 15:...	0000-00-00 00:...	61.99.183.91	n
<input type="checkbox"/>	135	391	00B7EF8D-D42...	2011-11-29 14:...	2011-11-29 15:...	211.178.131.12	y



Demo

Demo

<http://github.com/jeongjuwon/planche>

Q & A

<https://www.facebook.com/groups/korea.sencha/>