

오픈소스 개발자 입문

- CUBRID를 통해 간접 경험하기 -

민 준 / 오픈프론티어 3기

2016. 11. 10



CONTENTS

- 오늘 이야기하고 싶은건?
- 나의 이야기
 - 나는 왜 오픈소스 개발자가 되었는가?
 - 막연한 출발점 ... 그리고 오픈프론티어
 - 오픈소스 개발자 되는법
- CUBRID를 통해 간접 경험 얻기
 - CUBRID란?
 - CUBRID 소스 관리 구조
 - CUBRID에 기여(contribution)하기
 - 올해 실제 기여한 부분
- 더 해주고 싶은 이야기
 - 제일 좋아하는, 제일 잘 아는 분야에서 시작하세요.

오늘 이야기하고 싶은건?



라이선스? 오픈소스 정신?

Github?

1년차 오픈소스 개발자 경험 공유

- 막연함 제거
- 친숙해지기
- 뻔하지만 도움이 되는 이야기

나는 왜 오픈소스 개발자가 되었나?

DBMS 개발자
(박성)

방향성(고민)



소통, 발전

DBMS 개발
(시스템 소프트웨어)

막연한 출발점 ... 그리고 오픈프론티어

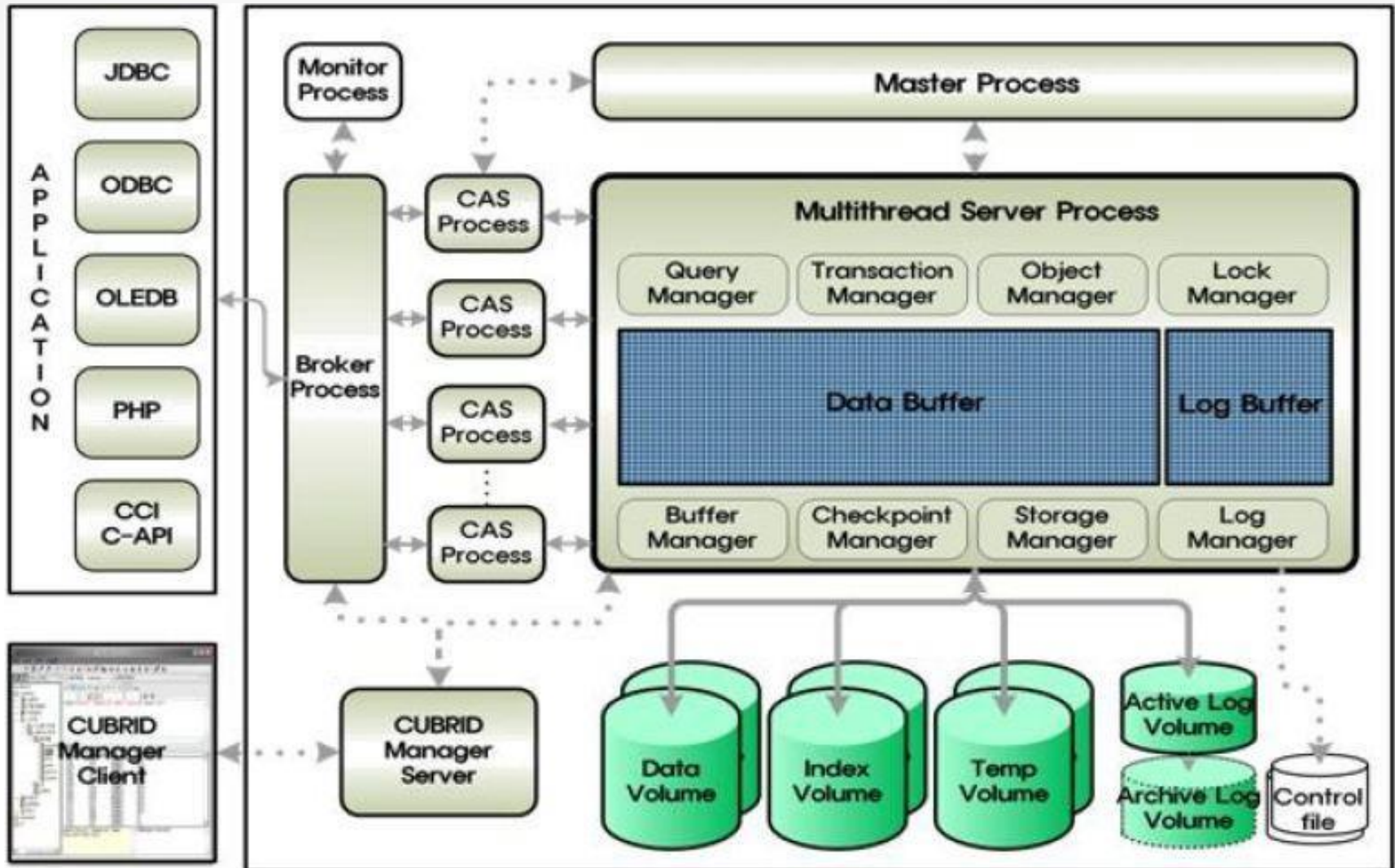
GitHub



오픈소스 개발자 되는법

- 존재하는 오픈소스에 기여
 - 오늘 이야기할 부분
 - CUBRID, LINUX kernel, Android
- 개인 아이디어 실현
 - 기존 오픈소스에 기반하여
 - 새로운 소프트웨어 (BIGDATA 관련)

CUBRID 란?



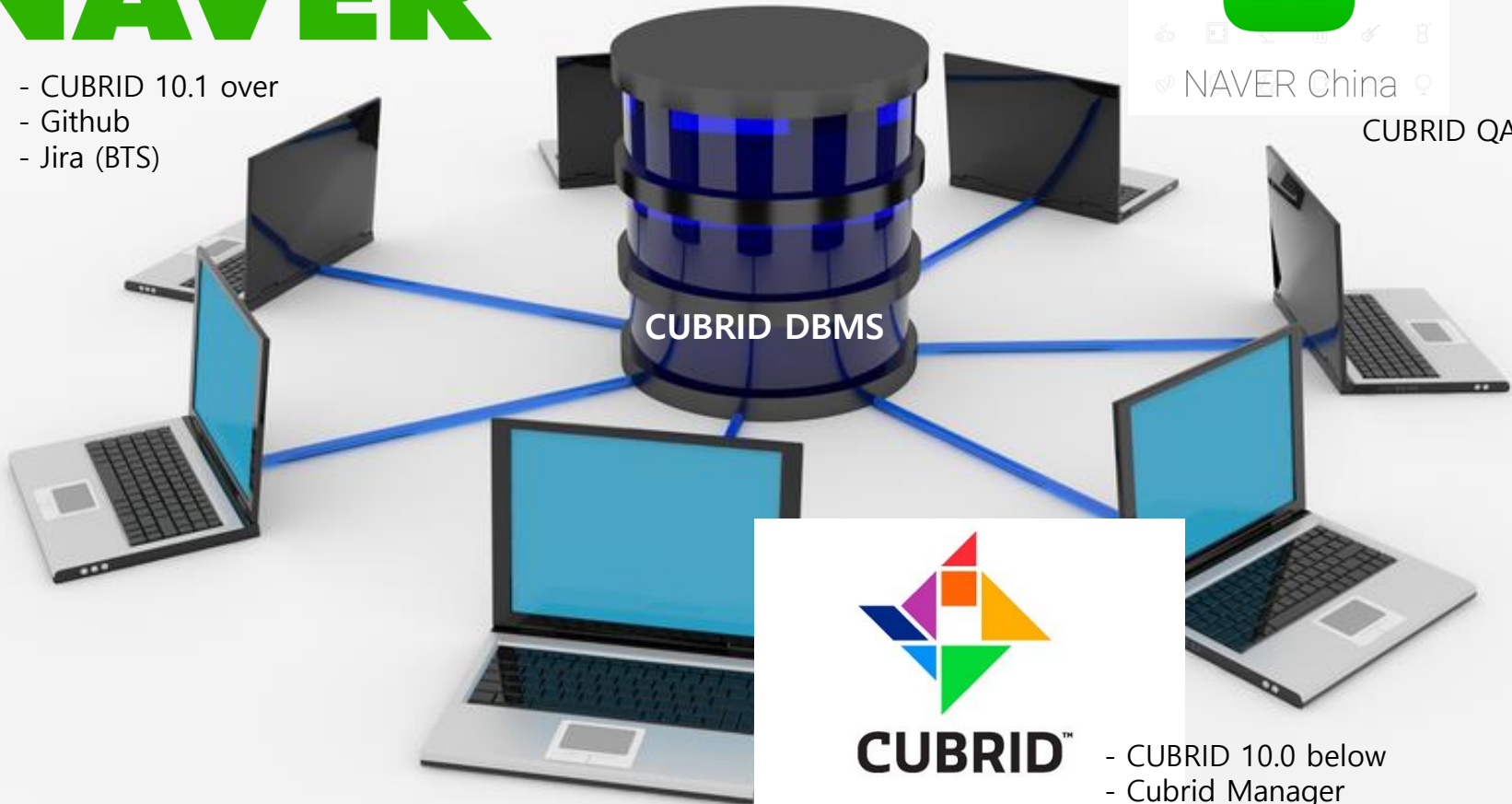
CUBRID 소스 관리 구조

NAVER

- CUBRID 10.1 over
- Github
- Jira (BTS)



CUBRID QA



CUBRID™

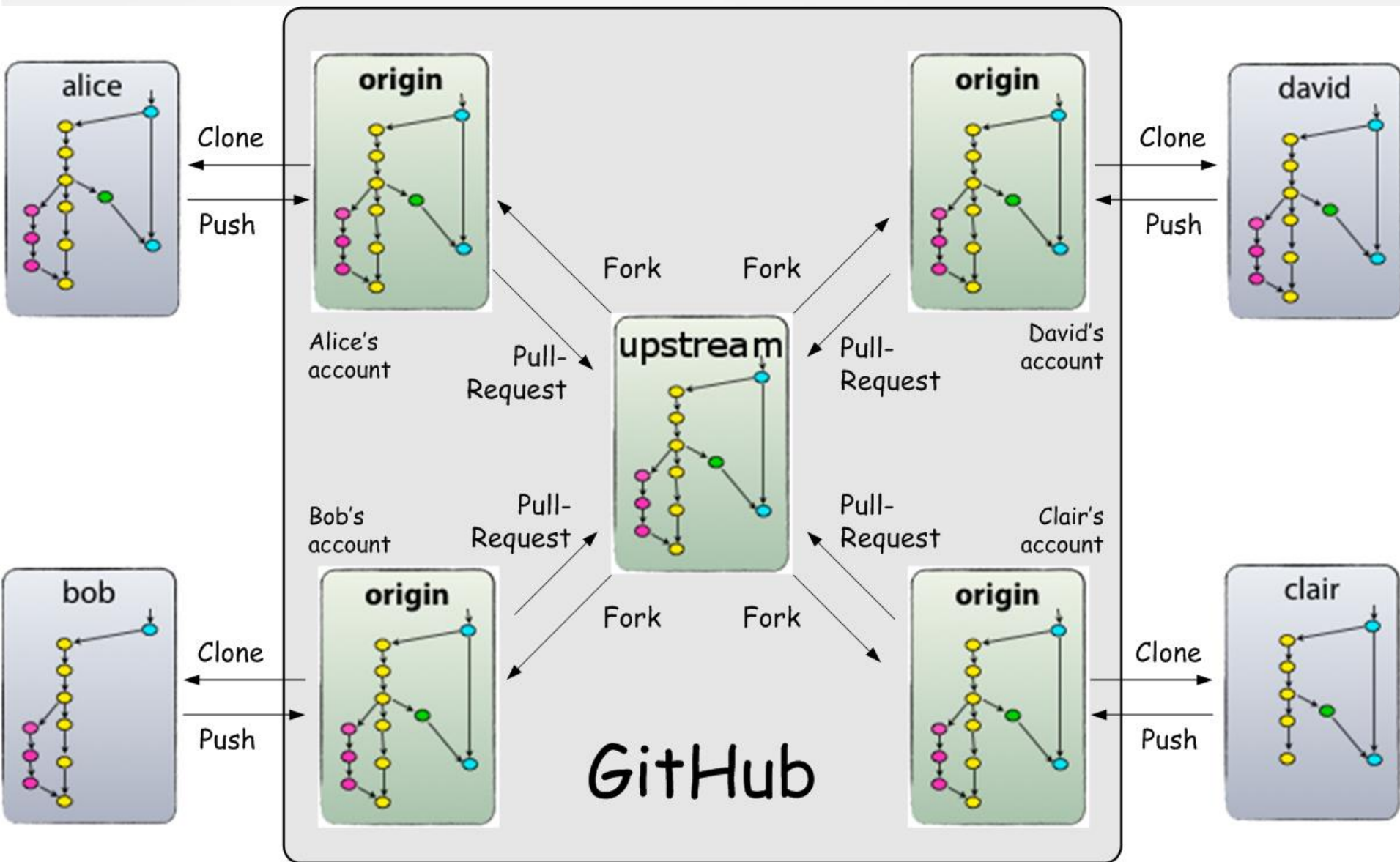
- CUBRID 10.0 below
- Cubrid Manager
- Drivers
- etc ...

WHO is involved

No	Person	Role or Responsibility	Location
1	Project Manager (committer)	<ul style="list-style-type: none">• Jira issue Management/Control• Jira Issue Assign to Person• CUBRID GitHub Managements• Approve PR• Control/Comments Developer/QA works	NAVER
2	CUBRID Manager	<ul style="list-style-type: none">• Developer Control in CUBRID, Coordination• Distribute OPEN works to Developer• Merge Approved PR	CUBRID
3	Developer	<ul style="list-style-type: none">• Create Jira Issues• Fork GitHub Repository• Clone to Local Repository• Source Change in Local• Commit to Local Repository• Push to forked GitHub Repository• Pull Request• Change status of Jira• Creates Test Cases for modified feature	CUBRID, NAVER and anywhere
4	CUBRID QA	<ul style="list-style-type: none">• DailyQA for modified sources	CUBRID
5	QA	<ul style="list-style-type: none">• Final QA• Close Jira Issue	NAVER China

Definitions

- Jira: Naver가 운영하는 CUBRID issue tracker
 - Issue의 생성/담당자 지정/종결까지의 모든 workflow
 - Naver의 '관리자'가 총괄하고 있음.
 - GitHub와 연동됨
 - GitHub 작업 전에 Jira Issue가 먼저 생성되어야함
(Jira issue 없는 PR등은 의미 없음)
 - 모든 작업의 시작과 종착점
 - Email은 communication의 보조 수단이며 tracking이 가능한 jira 사용 권고
(Jira/GitHub의 Comment 활용하여 History를 남기는 의미가 있음)
- GitHub (CUBRID)
 - 변형된 “Vincent Drissen Model”을 적용.
 - 'develop' branch에서 작업.
 - ZAB(Zero Activity Bug) 상태가 되었을 때 'release' branch로 move.
 - Jira와 연동됨.



CUBRID에 기여하기

1. CUBRID Jira에 이슈 등록 (<http://jira.cubrid.org>)
2. 관리자 승인 및 이슈 배정
3. 이슈 담당자에 의한 CUBRID Jira 이슈 상태 변경 'IN Progress'
4. GitHub에서 소스코드 Fork – Clone – Create branch
5. source 수정 작업
6. Commit (git commit -m "[CBRD-20355] Create Default Extension")
7. Push
8. Pull Request
9. Committer review (5) 수정 ~ (8) Pull Request 반복
10. Merge Commits (using 'Squash merge'), CUBRID 관리자에게만 권한이 있음.
11. QA
12. CUBRID Jira 상태 종료

Stage1: CUBRID JIRA Issue Creation


1. Motivation: 10.1에 포함되어야 하는 기능 개선/Fix등
2. Issue에 대한 공감 과정: 구두 또는 email등으로 issue에 대한 필요성을 NAVER의 Project Manager와 서로 공감할 필요가 있음
3. Issue 등록
 - 1) Jira에 Issue Creation
 - GitHub 등에서 작업을 하기 전에 이 작업에 대한 issue를 먼저 jira에 등록하여야 한다. GitHub도 jira와 연동되어 있기 때문에 이 과정이 작업의 시작점 이다.
 - Issue에 대한 통제권은 naver가 가지고 있다

JIRA Issue 등록 화면

Create Issue

 Configure Fields ▼

Project*  CUBRID (CBRD) ▼

Issue Type* ☒ Correct Error ▼ 

Some issue types are unavailable due to incompatible field configuration and/or workflow associations.

Summary*

Priority*  Minor ▼ 

Component/s*

Start typing to get a list of possible matches or press down to select.

Description Style ▼ B I U A ↻A ▼  ▼  ▼    ▼ + ⌵



client crashes when a bad SQL hint is given to UPDATE/DELETE



Edit



Comment

Assign

More ▾

Reopen Bug

Description

SYMPTOM

- client crashes when a bad SQL hint is given to UPDATE/DELETE statement

TEST VERSION

```
$ cubrid_rel
```

```
CUBRID 10.1 (10.1.0.7194-65c6815) (64bit release build for Linux) (Nov 1 2016 09:50:48)
```

REPRODUCE STEP

```
drop table if exists foo;  
create table foo (a int );
```

```
DELETE /*+ use_nl(bad_hint) */ FROM foo;  
UPDATE /*+ use_nl(bad_hint) */ foo SET a = 1;
```

CALLSTACK

```
Program terminated with signal 11, Segmentation fault.  
#0 0x00007f3a3105b149 in pt_resolve_hint (Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.192.el6.x86_64 libgcc-4.4.7-17.el6.x86_64  
libstdc++-4.4.7-17.el6.x86_64 ncurses-libs-5.7-4.20090207.el6.x86_64  
---Type <return> to continue, or q <return> to quit---  
(gdb) bt  
#0 0x00007f3a3105b149 in pt_resolve_hint (parser=0x811f70, node=<value optimized out>) at /home/hornetmj/work/CUBRID_10/src/parser/name_resolution.c:6628  
#1 0x00007f3a3105bbaa in pt_bind_names (parser=0x811f70, node=0x819598, arg=0x7ffdf10cd20, continue_walk=0x7ffdf10cd00)  
at /home/hornetmj/work/CUBRID_10/src/parser/name_resolution.c:2668  
#2 0x00007f3a31064e30 in pt_walk_private (parser=0x811f70, node=0x819598, void_arg=0x7ffdf10cce0)  
at /home/hornetmj/work/CUBRID_10/src/parser/parse_tree_cl.c:918  
#3 0x00007f3a31065068 in parser_walk_tree (parser=<value optimized out>, node=<value optimized out>, pre_function=<value optimized out>,  
pre_argument=<value optimized out>, post_function=<value optimized out>, post_argument=<value optimized out>)  
at /home/hornetmj/work/CUBRID_10/src/parser/parse_tree_cl.c:918  
#4 0x00007f3a3105441f in pt_resolve_names (parser=0x811f70, statement=0x819598, sc_info=<value optimized out>)  
at /home/hornetmj/work/CUBRID_10/src/parser/name_resolution.c:7785  
#5 0x00007f3a3109b244 in pt_check_with_info (parser=0x811f70, node=0x819598) at /home/hornetmj/work/CUBRID_10/src/parser/semantic_check.c:10592  
#6 pt_semantic_check (parser=0x811f70, node=0x819598) at /home/hornetmj/work/CUBRID_10/src/parser/semantic_check.c:10937  
#7 0x00007f3a3104e236 in pt_compile (parser=0x811f70, statement=0x819598) at /home/hornetmj/work/CUBRID_10/src/parser/compile.c:394  
#8 0x00007f3a30f9e47b in db_compile_statement_local (session=<value optimized out>) at /home/hornetmj/work/CUBRID_10/src/compat/db_vdb.c:597  
#9 0x00007f3a30f491dc in csql_execute_statements (csql_arg=0x7ffdf1133f0, type=<value optimized out>, stream=<value optimized out>, line_no=-1)  
at /home/hornetmj/work/CUBRID_10/src/executables/csql.c:1834  
#10 0x00007f3a30f4aa93 in csql_do_session_cmd (csql_arg=0x7ffdf1133f0) at /home/hornetmj/work/CUBRID_10/src/executables/csql.c:900  
#11 start_csql (csql_arg=0x7ffdf1133f0) at /home/hornetmj/work/CUBRID_10/src/executables/csql.c:664
```

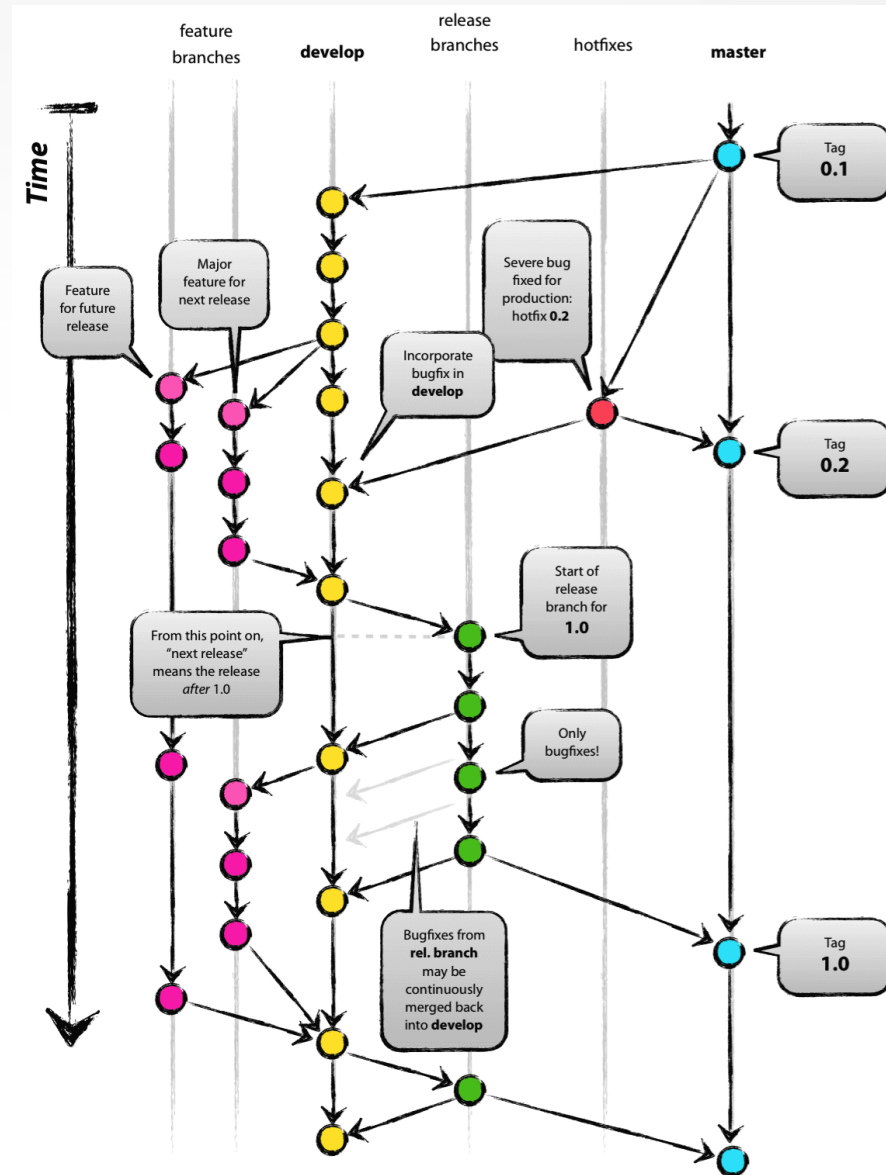
JIRA Issue 등록 유의 사항

1. 영어로 작성하여야 한다.
2. 작성해야 할 주요 항목 (*표)
 - 1) Project (아래중 택1, 개발이라면 기본인 'CUBRID (CBRD)' 선택)
 - CUBRID (CBRD)
 - CUBRIDQA
 - 서스테인 플랫폼
 - 2) Issue Type (택1, 개발과제라면 'improve function/performance')
 - Correct Error
 - Improve Function/Performance
 - 3) Summary
 - 한줄 정도의 문장으로 기술한다. 이 summary는 이 issue가 끝날때까지 제목으로 사용될 것이기 때문에 최대한 간결하면서 하는 일이 무엇인지 기술하도록 한다.
 - 4) Priority: 'Minor' 선택 (Major/Critical로 해도 naver에서 변경한다)
 - 5) Component: 적당한 것이 없으면 CUBRID (이런 부분도 naver에서 수정함)
 - 6) Description:
 - 문제가 무엇이고 (problem definition)
 - 개발시 예상되는 문제점 (side effect)
 - 개발 전략/방법
등을 기술한다. (다른 사람것을 참조하고, 미리 TEXT 형태로 만들고 가능하면 review 받으면 좋다)
 - 7) Original estimate: 예상 시간 기술(꼭 필요하지는 않는듯)
3. Assignee등의 항목은 그대로 두고 'CREATE' 하면 됨. (naver 관리자가 다 변경해 준다.)
4. CREATE가 완료되면 [CBRD-12345] 형태의 번호가 생성되는데, 이 번호를 추후 Git Commit의 Comment등에 사용.
(종결될 때까지)

Stage2: Clone a Repository

1. Jira에서 ISSUE가 NAVER 관리자에 의해서 'Confirmed' 되면
2. Jira 해당 ISSUE에서 "Start Work"를 선택하여 "IN PROGRESS"로 상태를 바꾼다.
3. (GitHub에서) 작업할 branch를 fork한다.
4. Fork한 Git Repository를 Clone 한다.
 - 1) (Git Hub) GitHub에서 Clone할 branch를 선택후 'Clone or download' 선택
 - 2) (Local) git clone [git@github.com:hor-net-mj/cubrid.git](https://github.com/hor-net-mj/cubrid.git)
5. Feature Branch를 만든다.
 - 1) 향후의 Pull Request는 이 feature branch 이름이 키가 되기 때문에 feature branch를 만들고 여기에서 작업
 - 2) (Local) git checkout -b 'featurea명' develop(예: git checkout -b create_default develop)
 - 3) (Local) git branch (현재 branch가 초록색으로 highlight)

Vincent Drissen Model



Stage3: Source 수정

1. CUBRID Coding Convention을 따른다. (http://www.cubrid.org/coding_conventions)
2. Short Guideline
 - 1) 각 LINE이 120 column을 넘지 않도록 한다.(comment, code)
 - 2) 'indent' 명령으로 수정한다.
(indent -l120 -lc120 {filename})
3. 기존 소스 분석에 의한 know-how
 - 1) Indentation은 2개의 blank로 한다.
 - 2) 모든 if는 하나의 문장이라도 block 처리 한다.
틀린예 : if (x == 1)
 y++;
바른예: if (x == 1)
 {
 y++;
 }
 - 3) if-else
 - a. if 다음에 한칸 띄우고 '(' condition ')'. 예: if (x == 2 && y = 1)
 - b. 다음줄에는 2칸 띄우고 '{'
 - c. 다음줄에 '{' 뒤에 두칸 띄우고 code
 - d. '{'에 맞추어 '}'

```
if (point == 1)
{
    scan_status = SUCCESS;
}
else
{
    scan_status = FAIL;
}
```

Stage3: Source 수정

4) Switch

- a. switch 한칸 띄우고 (condition)
- b. 두칸 띄우고 {
- c. 위의 '{'에 맞추어 case RT:
- d. Case 아래 두칸 띄고 statement

```
switch (type)
```

```
{
```

```
case READ:
```

```
    index = 0;
```

```
    break;
```

```
case WRITE:
```

```
    index = 1;
```

```
default:
```

```
    break;
```

```
}
```


Stage4: Commit to Local Repository /Push to GitHub(fork)

1. Commit 준비

- 1) `git status` (명령을 실행하면 수정된 부분이 빨간색으로 표시된다.)
- 2) `git add`: Commit에 포함할 파일에 대해서: `git add {filename}`
- 3) `git status`하면 add된 FILE이 초록색으로 Highlight 된다.
- 4) 제외할 FILE은: '`git checkout - {filename}`' 하면 수정전으로 변경된다.
예)

```
git status
git add query_dump.c
git status
git checkout -- parser/csql_grammar.c
```

2. Commit

- JIRA의 ISSUE 번호 사용을 사용해야 된다.
 - 1) JIRA와 GitHub가 연동되기 때문에 **JIRA 번호를** 써줘야 한다
 - Commit Message: [CBRD-<issue-number> description
 - 예:

`commit -m "[CBRD-20355] CREATE DEFAULT Extension"`

- Commit을 하면 'change'가 local repository에 반영된다.(local의 소스 수정본)

3. GitHub Fork본에 수정내용 PUSH (feature branch를 이용하여)

- `push origin {feature branch명}`
(예:

```
push origin create_default
```

이러면 GitHub fork본에 '**create_default**' branch가 생긴다)

Stage5: Pull Request (GitHub)

1. GitHub에 접속한다.
2. Fork본으로 page 변경한다.

The screenshot shows the GitHub interface for a repository named 'kisoo-han / cubrid', which is a fork of 'CUBRID/cubrid'. The repository has 1 watch, 0 stars, and 27 forks. The main navigation bar includes links for Code, Pull requests (0), Projects (0), Wiki, Pulse, Graphs, and Settings. A description of CUBRID as an open-source relational database management system is provided, along with the website URL 'http://www.cubrid.org' and an 'Edit' link. A blue arrow points to the '7,171 commits' link. Below this, a progress bar shows the repository's status relative to the 'develop' branch, indicating it is 2 commits ahead and 7 commits behind. A table of recent commits is visible, including one by 'kisoo-han' extending the DEFAULT clause in CREATE statements and another by 'broker' fixing GCC5 compatibility issues.

kisoo-han / cubrid
forked from CUBRID/cubrid

Unwatch 1 Star 0 Fork 27

Code Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

CUBRID is a comprehensive open source relational database management system highly optimized for Web Applications.
<http://www.cubrid.org> — Edit

7,171 commits 5 branches 0 releases 14 contributors

Branch: develop New pull request Create new file Upload files Find file Clone or download

This branch is 2 commits ahead, 7 commits behind CUBRID:develop. Pull request Compare

kisoo-han [CBRD-20399]	Extension of DEFAULT Clause in CREATE statements	Latest commit 22df509 7 days ago
broker	[CBRD-20453] Fixing GCC5 compatibility issues (#201)	2 months ago

Stage5: Pull Request (GitHub)

1. Branch를 feature branch(예: create_default) 로 변경한다.
2. PULL Request 시행

The screenshot shows the GitHub interface for the 'kisoo-han / cubrid' repository. At the top, it indicates the repository is forked from 'CUBRID/cubrid'. The repository has 1 star, 0 forks, and 27 watchers. The main navigation bar includes links for Code, Pull requests (0), Projects (0), Wiki, Pulse, Graphs, and Settings. A description of CUBRID is provided, along with a link to the website. A blue arrow points to the 'Branch: create_default' dropdown menu. Below this, there are buttons for 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A status bar indicates the current branch is 5 commits ahead and 7 commits behind the upstream 'develop' branch. A table of recent commits is shown, including the latest commit by 'kisoo-han' and several other commits related to fixing GCC5 compatibility and packaging issues.

kisoo-han / cubrid
forked from CUBRID/cubrid

Unwatch 1 Star 0 Fork 27






Code Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

CUBRID is a comprehensive open source relational database management system highly optimized for Web Applications.
<http://www.cubrid.org> — Edit

7,174 commits 5 branches 0 releases 14 contributors

Branch: create_default New pull request Create new file Upload files Find file Clone or download

This branch is 5 commits ahead, 7 commits behind CUBRID:develop. Pull request Compare

 kisoo-han [CBRD-20399] Extension of DEFAULT Clause in CREATE statements	Latest commit fa3c145 4 days ago
 broker [CBRD-20453] Fixing GCC5 compatibility issues (#201)	2 months ago
 cas [CBRD-20155] add packaging feature for CMake and remove files of old ...	4 months ago
 cci [CBRD-20323] revert off_t type in system.h for WINDOWS (#118)	4 months ago
 cm_common [CBRD-20155] add packaging feature for CMake and remove files of old ...	4 months ago

Stage5: Pull Request

1. 하나의 feature branch에 대해서 여러 개의 PULL Request를 하면,
2. Branch 이름에 대해서 하나의 PULL Request에 여러 개의 Commit이 묶여 진다.
3. 따라서 두 건의 서로 다른 ISSUE의 pull request가 같은 이름의 branch ('develop')을 썼다면 하나의 PR로 묶임.
(하나의 PULL Request에 여러 개의 ISSUE가 섞이면 안된다)
4. 이런 이유로 반드시 feature branch를 사용하여 작업하고 commit/push/PR 하여야 함.

[CBRD-20399] Extension of DEFAULT Clause in CREATE statements #277

This is part of CUBRID modification to support TO_CHAR in default clause of CREATE statements. It consis...

[View pull request](#)





5 commits

12 files changed


4 commit comments




1 contributor

Commits on Oct 05, 2016

  **kisoo-han** [CBRD-20399] Extension of DEFAULT Clause in CREATE statements   28c09bd

Commits on Oct 06, 2016

  **kisoo-han** [CBRD-20399] Extension of DEFAULT Clause in CREATE statements  d230d63

  **kisoo-han** [CBRD-20399] Extension of DEFAULT Clause in CREATE statements  7f75426

Commits on Oct 07, 2016

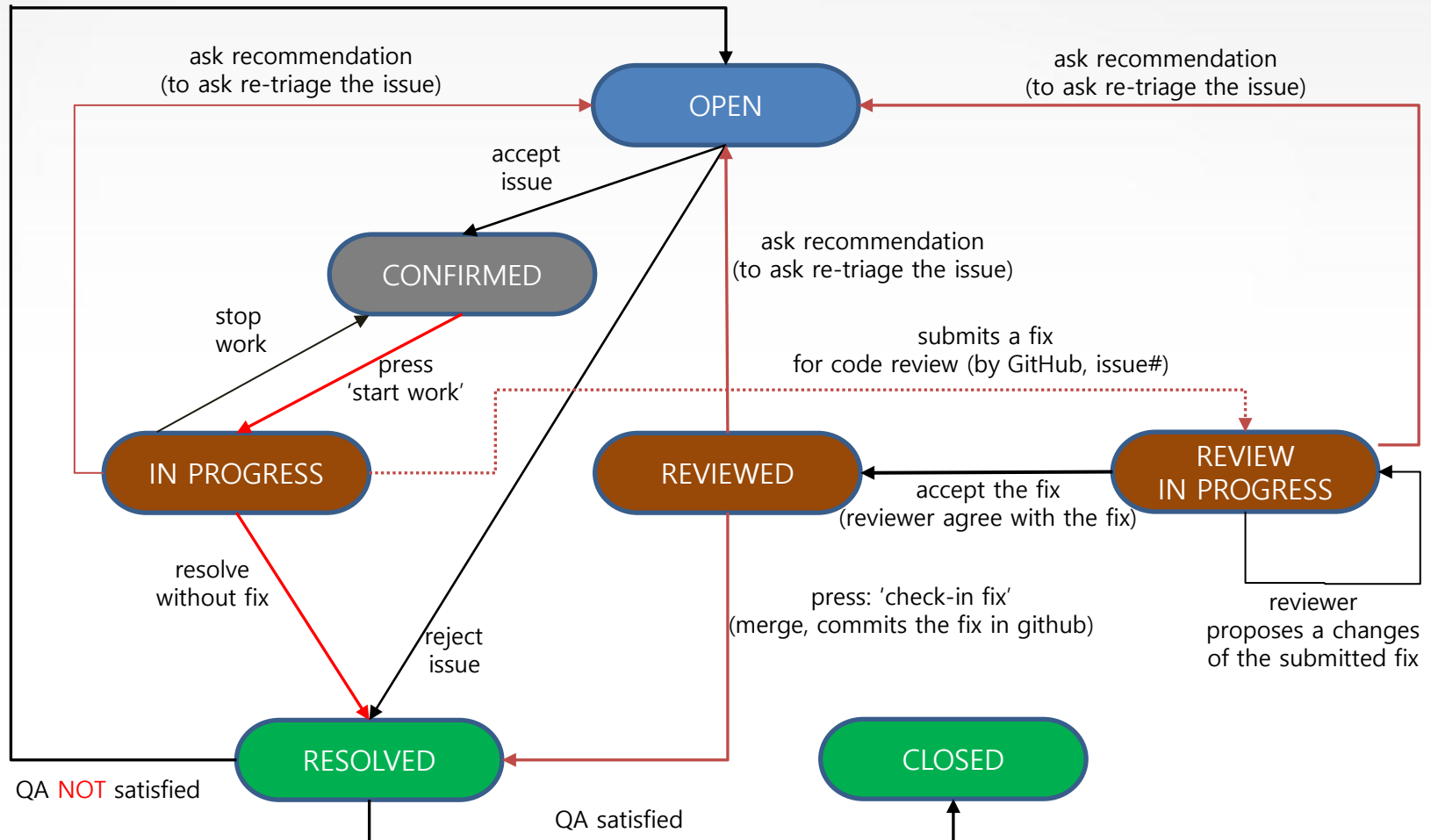
  **kisoo-han** [CBRD-20399] Extension of DEFAULT Clause in CREATE statements   88a275d

  **kisoo-han** [CBRD-20399] Extension of DEFAULT Clause in CREATE statements   fa3c145

Stage6: Review 과정

1. PULL Request가 되면, PR이 commit될때 사용했던 comment의 JIRA 번호로 notification이 간다.
2. 해당 JIRA의 상태가 'IN PROGRESS'에서 'REVIEW IN PROGRESS'로 자동 변경되며, 따라서 이 issue에 있는 watcher에게 mail등으로 통보되고 REVIEW가 시작된다.
3. Reviewer에 의해 소스가 수정되면
 1. 수정
 2. Git add
 3. Git commit
 4. Git push
 5. Pull request를 반복한다.
4. PUSH/PULL Request를 할 때는 feature branch에 유의해서 한다.
5. (1 ~ 4)의 과정이 반복되고 NAVER Project Manager에서 최종적으로 이 PULL Request가 'Approve' 되었다는 메일이 온다.
6. GitHub에서 관련된 PULL Request를 MERGE 한다. (이때 반드시 'Squash merge'를 사용하여 MERGE한다)
 - 일반 Merge는 개별 Commit들을 그대로 Merge한다.
 - 반면 'Squash merge'는 여러 개의 Commit을 하나의 Commit으로 합친후 Merge하여 HISTORY 가 간단해 진다.
(앞의 예에서도 PR하나에 5개의 commit이 있는데 'Squash merge'는 이걸 하나의 commit으로 만든후 Merge한다)
7. 변경이 GitHub에 반영되어었으니 JIRA의 해당 Page에서 'check-in fix'를 선택하여 상태를 'Resolved'로 변경한다. (**개발자 역할은 여기서 끝**)
8. QA를 실행하고 이 ISSUE를 Close하는 것은 NAVER에서 알아서 한다.

JIRA Workflow



실 사례 - 매뉴얼 기여

```
2 en/admin/config.rst
```

@@ -239,7 +239,7 @@ On the below table, if "Applied" is "server parameter", that parameter affects t

239	239		+	-----+	-----+	-----+	-----+	-----+	-----+
240	240			only_full_group_by		client parameter		0	bool
241	241	no		available					
242	242	-		oracle_style_empty_string		client parameter			bool
									no
	242	+		oracle_style_empty_string		client/server parameter			bool
									no
243	243		+	-----+	-----+	-----+	-----+	-----+	-----+
244	244			pipes_as_concat		client parameter			bool
245	245	yes							
			+	-----+	-----+	-----+	-----+	-----+	-----+

실 사례 – bug fix

9		src/parser/method_transform.c	View
		@@ -1162,7 +1162,14 @@ with_translate_spec (PARSER_CONTEXT * parser, PT_NODE * spec, void *void_arg,	
		select Sp_int(t3.nu_id), t3.nu_id, Sp_int(t3.nu_id), Sp_int2(t3.nu_id), Sp_string2()	
1162	1162	from tbl_1 t1	
1163	1163	if (dummy_set_tbl)	
1164	1164	inner-join tbl_2 t2 on t1.id=t2.id	
1165	1165	left outer join tbl_3 t3 on t1.id=t3.id	
		- spec->next = new_spec;	
		left outer join tbl_4 t4 on t1.id=t4.id	
	1165	+ /* We attach the new_spec to the end of the from list. */	
	1166	+ new_spec->next = NULL;	
	1167	+ for (tmp = spec; tmp->next != NULL; tmp = tmp->next)	
	1168	+ {	
	1169	+ left outer join tbl_8 t8 on t1.id=t8.id	
	1170	+ /*No body.*/	
	1171	+ left outer join tbl_9 t9 on t1.id=t9.id	
	1172	+ left outer join tbl_10 t10 on t1.id=t10.id	
		- tmp->next = new_spec;	
1166	1173	+ left outer join tbl_11 t11 on t1.id=t11.id	
1167	1174	+ left outer join tbl_12 t12 on t1.id=t12.id	
1168	1175	where t1.src_name='TA1';	
		else	

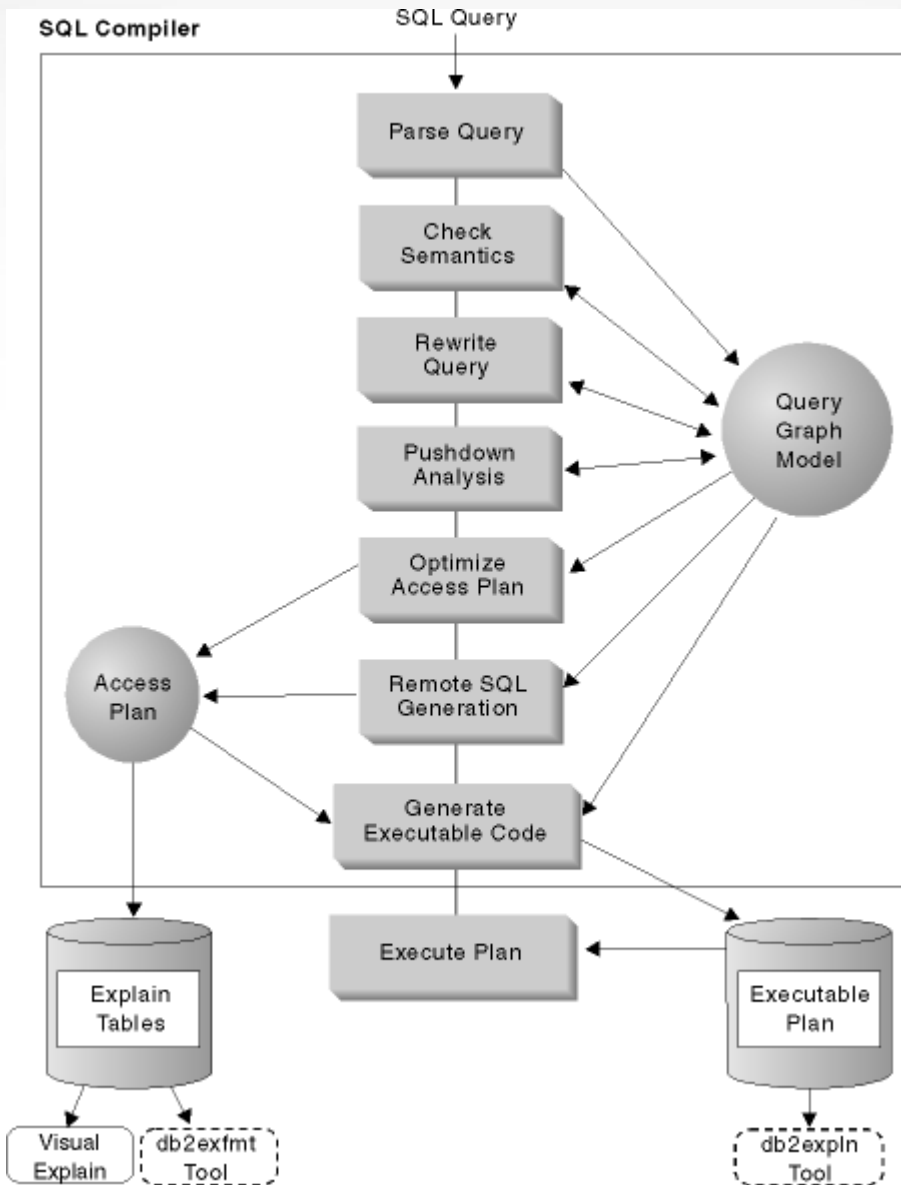
실 사례 – bug fix

```
6614 - /* clear hint info */
6615 - node->info.query.q.select.hint = PT_HINT_NONE;
6616 - if (*ordered != NULL)
6617 - {
6618 -     parser_free_tree (parser, *ordered);
6619 - }
6620 - if (*use_n1 != NULL)
6621 + {
6622 -     parser_free_tree (parser, *use_n1);
6623 - }
6624 - if (*use_idx != NULL)
6625 - {
6626 -     parser_free_tree (parser, *use_idx);
6627 - }
6628 - if (*index_ss != NULL)
6629 - {
6630 -     parser_free_tree (parser, *index_ss);
6631 - }
6632 - if (*index_ls != NULL)
6633 - {
6634 -     parser_free_tree (parser, *index_ls);
6635 - }
6636 - if (*use_merge != NULL)
6637 + switch (node->node_type)
6638 + {
6639 +     case PT_SELECT:
6640 +         if (*index_ss != NULL)
6641 +         {
6642 +             parser_free_tree (parser, *index_ss);
6643 +         }
6644 +         if (*index_ls != NULL)
6645 +         {
6646 +             parser_free_tree (parser, *index_ls);
6647 +         }
6648 +     case PT_DELETE:
6649 +     case PT_UPDATE:
6650 +         if (*ordered != NULL)
6651 +         {
6652 +             parser_free_tree (parser, *ordered);
6653 +         }
6654 +         if (*use_n1 != NULL)
```

UPDATE /*+ use_n1(x y) */
athlete x
SET
gender = 'F'
WHERE exists (
SELECT
1
FROM
event y
WHERE
x.code = y.code)

더 해주고 싶은 이야기

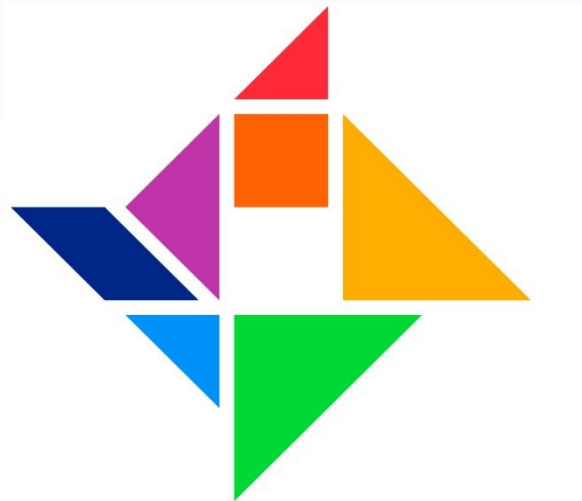
알고 있는 분야에 도전



더 해주고 싶은 이야기

정말 관심있거나 즐길수 있는 분야





CUBRID™