Install

```
[ExternalJoin-pgsql9.5.2]$ ls
db_install.sh    external_sample    ext_install.sh
install.sh    postgresql-9.5.2    table_init.sql

[ExternalJoin-pgsql9.5.2]$ sh install.sh
[ExternalJoin-pgsql9.5.2]$ ls
db_install.sh    external_sample    ext_install.sh
install.sh    postgresql-9.5.2    table_init.sql    tmp_install
```

DEMO

```
# at window[0]
[ExternalJoin-pgsql9.5.2]$ ./tmp_install/bin/postgres -D tmp_install/db/ -d 1
```

```
# at window[2]
[ExternalJoin-pgsql9.5.2]$ ./external_sample/join_sample
```

```
# at window[1]
[ExternalJoin-pgsql9.5.2]$ ./tmp_install/bin/psql
# execute following queries in psql

-- initialize random tables
DROP TABLE t1;
DROP TABLE t2;
CREATE TABLE t1(key int, dval double precision);
CREATE TABLE t2(key int, dval double precision);
-- insert sequential key and random generated value
INSERT INTO t1 (SELECT generate_series(1, 10000), random() * 100);
INSERT INTO t2 (SELECT generate_series(1, 10000),
   generate_series(1, 10) * 10 * random());

-- load module which sends tuples to external process
load 'external_join';
```

```
-- enable 'external_join' module
set external_join.enable = true;
-- execute external process here
-- send t1, t2 to outer process, WHERE clause is discarded
SELECT * FROM t1, t2 WHERE (t1.dval - t2.dval)^2 < 10;

-- disable 'external_join' module
set external_join.enable = true;
-- re-execute query on postgres:
-- verify the previous result by comparing it with this result
SELECT * FROM t1, t2 WHERE (t1.dval - t2.dval)^2 < 10;

-- OPTIONS --
-- address of the external process can be set like this
set external_join.addr = "127.0.0.1";
-- port of the external process can be also set
set external_join.port = 65535;
-- you can enable / disable of external join
set external_join.enable = true;
set external_join.enable = false;
```

NOTE

- This step must be executed in every postmaster (bin/postgres) call.

```
load 'external_join';
set external_join.enable = true;
```

- External process must be called before every SQL statement.
  - Since, this extension creates connection every SQL.

```
-- execute external process here
SELECT …;
```

Source

- Extension module is at postgresql-9.5.2/contrib/external_join
- Sample code for external process is at external_sample/*.cpp

Specifications

- Only primitive types (of C language) are supported.
  => array, variable length object are not supported.
- External process's tuple structure must match that of PostgreSQL.
  => See above 'Demo' and sample code external_sample/join_sample.cpp
  - External join module (this extension) sends
    1. Size of tuples
    2. Contents of tuples
  - External join module (this extension) receives
    - Contents of result tuples
    - When the connection is closed, this ends receiving.
- Size of a target relation must be small enough to fit in memory.
  - If the size is bigger than 1GB or 512MB or something, PostgreSQL cannot prepare buffer to hold the relation in sending phase.
- You cannot use 'Tab auto completion' because it is also SQL processing (and SQL processing is hooked here).

In case for freezing or deadlock

- This extension supports query cancel to some extent but freezing or deadlock occurs
  1. Query is cancelled when specific step of program.
  2. External process stops (freeze or deadlock), but it maintains connection.
     - This extension cannot determine whether result receiving ends.

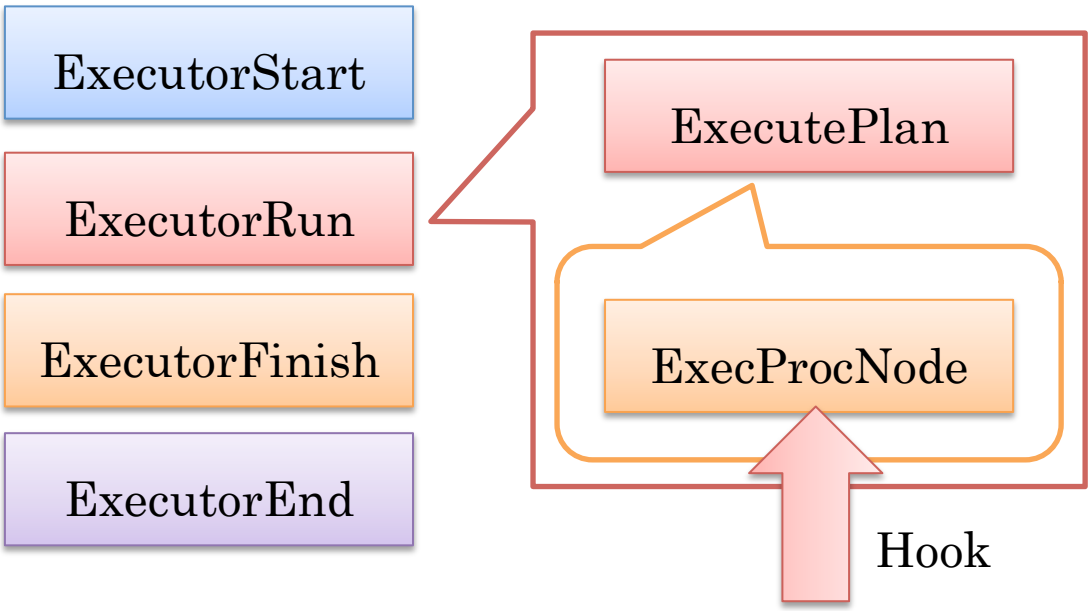
```
[ExternalJoin-pgsql9.5.2]$ ps ax | grep postgres
24824   ??  Ss     0:00.00 postgres: checkpointer process
24828   ??  Ss     0:00.00 postgres: stats collector process
24830   ??  Ss     0:41.40 postgres: Username Username [local] SELECT
24822 s004  S+     0:00.02 ./tmp_install/bin/postgres -D tmp_install/db/ -d 1
24835 s006  S+     0:00.00 grep postgres
[ExternalJoin-pgsql9.5.2]$ kill -9 24830
```

- Executor -

- SQL プロセッサにおけるプランを実行する部分
  (@ backend/executor/execMain.c)
- Start/Run/Finish/End から成る
- 上記のそれぞれについてフックポイントが存在
  - フック: 呼び出す関数をすり替えること

・ 今回、PostgreSQL が用意しているフックポイントよりも細かい部分に
フックポイント(ExecProcNode_hook)を作成しフックした.
[ExecutorRun() -> ExecutePlan() -> ExecProcNode()]
  - ExecProcNode(): プランツリーを実行し, 結果タプルを返す関数.

・ ExecProcNode_hook で ExecProcNode()をフックし, プランツリーの実行と結果タプルの返却を肩代わりする.



フックポイントの作成にあたって変更した部分

・ backend/executor/execMain.c [2 行]
・ include/executor/executor.h [2 行]
・ 外部リンケージを持った関数ポインタ変数を設定するだけ