



v9.5の新機能

Custom Scan/Join Interface

KaiGai Kohei <kaigai@kaigai.gr.jp>

v9.5 New Feature

The screenshot shows the commitdiff page for commit 0b03e5. The title is "Introduce custom path and scan providers." The author is Robert Haas, and the committer is Robert Haas. The commit message states: "This allows extension modules to define their own methods for scanning a relation, and get the core code to use them. It's unclear as yet how much use this capability will find, but we won't find out if we never commit it." The commit is reviewed by KaiGai Kohei, Shigeru Hanada, Tom Lane, Andres Freund, Alvaro Herrera, and myself. A list of 22 files changed is shown, including src/backend/commands/explain.c, src/backend/executor/Makefile, src/backend/executor/execAmi.c, src/backend/executor/execProcnode.c, src/backend/executor/nodeCustom.c (new file with mode: 0644), src/backend/nodes/copyfuncs.c, src/backend/nodes/outfuncs.c, src/backend/optimizer/planmain.c, src/backend/optimizer/pathnode.h, src/include/nodes/plannodes.h, src/include/nodes/relation.h, src/include/optimizer/pathnode.h, and src/include/optimizer/planmain.h. A large oval highlights the title "Custom-Scan Interface".

projects / postgresql.git / commitdiff

summary | [shortlog](#) | [log](#) | [commit](#) | [tree](#) | [raw](#) | [patch](#) (parent: 7250d85)

commit search: re

Introduce custom path and scan providers.

author Robert Haas <rhaas@postgresql.org>
Fri, 7 Nov 2014 22:28:02 +0000 (17:28 -0500)
committer Robert Haas <rhaas@postgresql.org>
Fri, 7 Nov 2014 22:34:38 +0000 (17:34 -0500)

This allows extension modules to define their own methods for scanning a relation, and get the core code to use them. It's unclear as yet how much use this capability will find, but we won't find out if we never commit it.

KaiGai Kohei, reviewed at various times and in various levels of detail by Shigeru Hanada, Tom Lane, Andres Freund, Alvaro Herrera, and myself.

22 files changed:

src/backend/commands/explain.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/executor/Makefile [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/executor/execAmi.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/executor/execProcnode.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/executor/nodeCustom.c [new file with mode: 0644] [patch](#) | [blob](#)
src/backend/nodes/copyfuncs.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/nodes/outfuncs.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/optimizer/planmain.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/optimizer/pathnode.h [patch](#) | [blob](#) | [blame](#) | [history](#)
src/include/nodes/plannodes.h [patch](#) | [blob](#) | [blame](#) | [history](#)
src/include/nodes/relation.h [patch](#) | [blob](#) | [blame](#) | [history](#)
src/include/optimizer/pathnode.h [patch](#) | [blob](#) | [blame](#) | [history](#)
src/include/optimizer/planmain.h [patch](#) | [blob](#) | [blame](#) | [history](#)

diff --git a/src/backend/commands/explain.c b/src/backend/commands/explain.c
index 3fcd1dd..99a0f0 100644 (file)

The screenshot shows the commitdiff page for commit e7cb7e. The title is "Allow FDWs and custom scan providers to replace joins with scans." The author is Robert Haas, and the committer is Robert Haas. The commit message states: "Foreign data wrappers can use this capability for so-called 'join pushdown': that is, instead of executing two separate foreign scans and then joining the results locally, they can generate a path which performs the join on the remote server and then is scanned locally. This commit does not extend postgres_fdw to take advantage of this capability; it just provides the infrastructure." The commit is reviewed by KaiGai Kohei, Shigeru Hanada, Ashutosh Bapat, and me. A list of 20 files changed is shown, including doc/src/sgml/custom-scan.sgml, doc/src/sgml/fdwhandler.sgml, src/backend/commands/explain.c, src/backend/executor/execAmi.c, src/backend/executor/execProcnode.c, src/backend/optimizer/planmain.c, src/backend/optimizer/pathnode.h, src/include/nodes/plannodes.h, src/include/nodes/relation.h, src/include/optimizer/pathnode.h, and src/include/optimizer/planmain.h. A large oval highlights the title "Custom-Join Interface".

projects / postgresql.git / commitdiff

summary | [shortlog](#) | [log](#) | [commit](#) | [tree](#) | [raw](#) | [patch](#) (parent: 2b22795)

commit search: re

Allow FDWs and custom scan providers to replace joins with scans.

author Robert Haas <rhaas@postgresql.org>
Fri, 1 May 2015 12:50:35 +0000 (08:50 -0400)
committer Robert Haas <rhaas@postgresql.org>
Fri, 1 May 2015 12:50:35 +0000 (08:50 -0400)

Foreign data wrappers can use this capability for so-called "join pushdown": that is, instead of executing two separate foreign scans and then joining the results locally, they can generate a path which performs the join on the remote server and then is scanned locally. This commit does not extend postgres_fdw to take advantage of this capability; it just provides the infrastructure.

Custom scan providers can use this in a similar way. Previously, it was only possible for a custom scan provider to scan a single relation. Now, it can scan an entire join tree, provided of course that it knows how to produce the same results that the join would have produced if executed normally.

KaiGai Kohei, reviewed by Shigeru Hanada, Ashutosh Bapat, and me.

20 files changed:

doc/src/sgml/custom-scan.sgml [patch](#) | [blob](#) | [blame](#) | [history](#)
doc/src/sgml/fdwhandler.sgml [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/commands/explain.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/executor/execAmi.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/executor/execProcnode.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/optimizer/planmain.c [patch](#) | [blob](#) | [blame](#) | [history](#)
src/backend/optimizer/pathnode.h [patch](#) | [blob](#) | [blame](#) | [history](#)
src/include/nodes/plannodes.h [patch](#) | [blob](#) | [blame](#) | [history](#)
src/include/nodes/relation.h [patch](#) | [blob](#) | [blame](#) | [history](#)
src/include/optimizer/pathnode.h [patch](#) | [blob](#) | [blame](#) | [history](#)
src/include/optimizer/planmain.h [patch](#) | [blob](#) | [blame](#) | [history](#)



Custom Scan/Join Interfaceが実現する事

- クエリ実行計画の一部を、拡張モジュール実装の“何か”で置き換える
- 何を置き換える事ができるのか？

- Scan ... set_rel_pathlist_hook で候補パス追加
- Join ... set_join_pathlist_hook で候補パス追加
- Others ... planner_hookを使ってインジェクションすれば何でもアリ。
→ 局所最適にならないよう、注意は必要だが....

■ 応用例

- GPUアクセラレーションによる全体的なRDBMS性能の底上げ。
→ PG-Strom (オタワで発表するんでヨロ)
- 特定のテーブル同士のJoinをMaterialized Viewスキャンに透過的に置換え。
- 特定のテーブルのうち、参照頻度の高い列だけを列指向形式でキャッシュ
- Scan/Joinの中間結果から統計情報を採取。値によるヒストグラム作成。
-など



クエリ実行計画

```
postgres=# EXPLAIN SELECT cat, AVG(x)
          FROM t0 NATURAL JOIN t1 RIGHT JOIN t2 ON t0.aid=t2.bid OR t0.bid=t2.bid
          WHERE atext like '%abc%' GROUP BY cat;
```

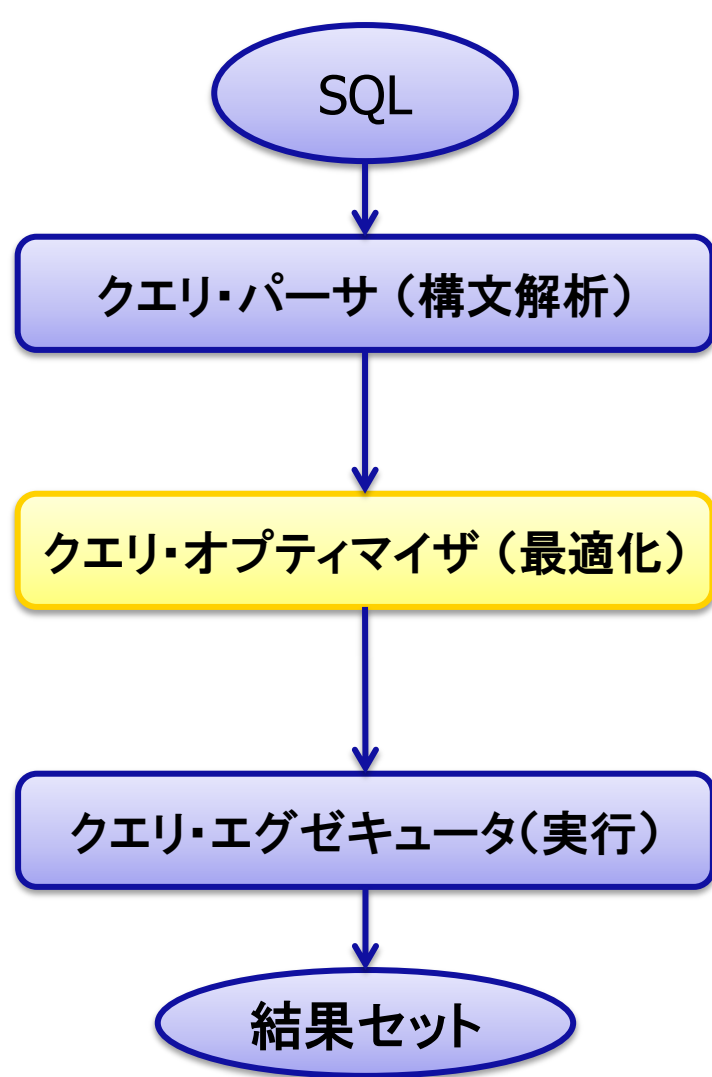
QUERY PLAN

```
-----
HashAggregate  (cost=979149.25..979149.58 rows=26 width=12)
Group Key: t0.cat
-> Nested Loop  (cost=834.05..979139.33 rows=1985 width=12)
    Join Filter: ((t0.aid = t2.bid) OR (t0.bid = t2.bid))
-> Seq Scan on t2  (cost=0.00..734.00 rows=40000 width=4)
-> Materialize  (cost=834.05..281207.82 rows=996 width=20)
    -> Hash Join  (cost=834.05..281202.84 rows=996 width=20)
        Hash Cond: (t0.aid = t1.aid)
        -> Seq Scan on t0  (cost=0.00..242858.60 rows=10000060 width=20)
        -> Hash  (cost=834.00..834.00 rows=4 width=4)
            -> Seq Scan on t1  (cost=0.00..834.00 rows=4 width=4)
                Filter: (atext ~~ '%abc%'::text)

(12 rows)
```



PostgreSQL クエリ処理の流れ (1/2)



リレーションに対する候補パスを列挙

- Base Relation
- Join Relation

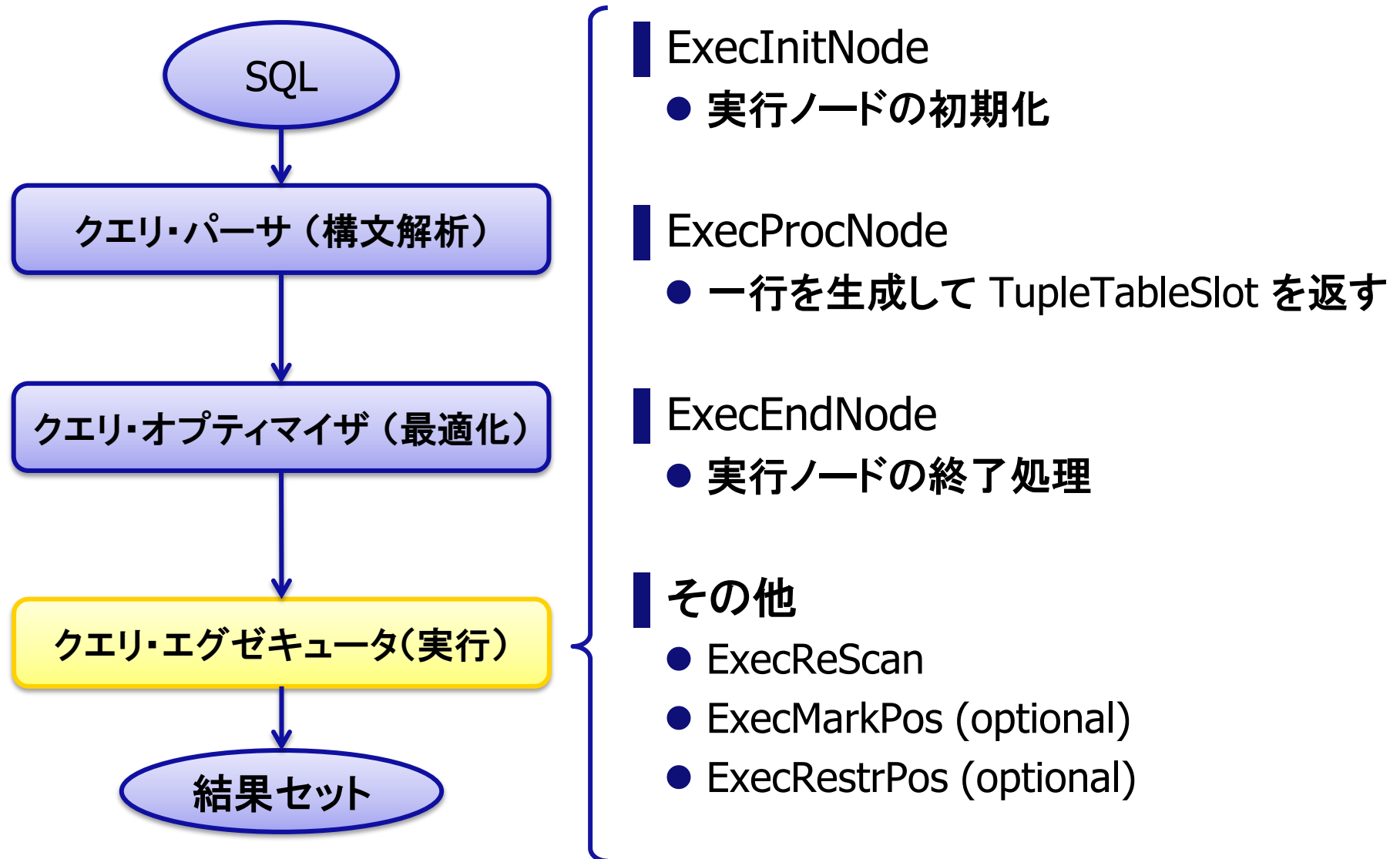
プラナーが最も実行コストの小さなパスを選択。

→ 全てのテーブルを単一のJoin-Treeに収めるまで結合順を試行錯誤

実行パスを元に、実行計画を作成



PostgreSQL クエリ処理の流れ (2/2)



Custom Scan/Joinを使うには (1/3)

候補パスを追加する

```
typedef void (*set_rel_pathlist_hook_type) (PlannerInfo *root,  
                                             RelOptInfo *rel,  
                                             Index rti,  
                                             RangeTblEntry *rte);  
  
extern PGDLLIMPORT set_rel_pathlist_hook_type set_rel_pathlist_hook;  
  
typedef void (*set_join_pathlist_hook_type) (PlannerInfo *root,  
                                              RelOptInfo *joinrel,  
                                              RelOptInfo *outerrel,  
                                              RelOptInfo *innerrel,  
                                              JoinType jointype,  
                                              JoinPathExtraData *extra);  
  
extern PGDLLIMPORT set_join_pathlist_hook_type set_join_pathlist_hook;
```

CustomPath 構造体

```
typedef struct CustomPath  
{  
    Path      path;                // Path 共通部分  
    uint32    flags;               // CUSTOMPATH_* flags  
    List      *custom_private;     // プライベート領域  
    const CustomPathMethods *methods; // コールバック関数  
} CustomPath;
```



Custom Scan/Joinを使うには (2/3)

CustomPath → CustomScan

- プラナーがCustomPathを選択
→ CustomScanノードを生成する
必要がある

どういう事か？

- CustomPathから生成されるのは、
CustomScanだけではない。(将来的に)
✓ CustomSort、CustomAgg、...
- プラン木はcopyObject()可能である事

→ GpuScanの例

```
/* Path information of GpuScan */
typedef struct {
    CustomPath  cpath;
    /* RestrictInfo run on host */
    List        *host_qual;
    /* RestrictInfo run on devices */
    List        *dev_qual;
} GpuScanPath;
```

CustomScan → CustomScanState

- エグゼキュータの開始時に、
CustomScanノードの内容に基づいて
CustomScanStateを初期化。

→ GpuScanの例 CustomScanStateを継承

```
typedef struct {
    GpuTaskState  gts;

    BlockNumber   curr_blknum;
    BlockNumber   last_blknum;
    HeapTupleData scan_tuple;
    List          *dev_qual;

    cl_uint       num_rechecked;
} GpuScanState;
```



Custom Scan/Joinを使うには (3/3)

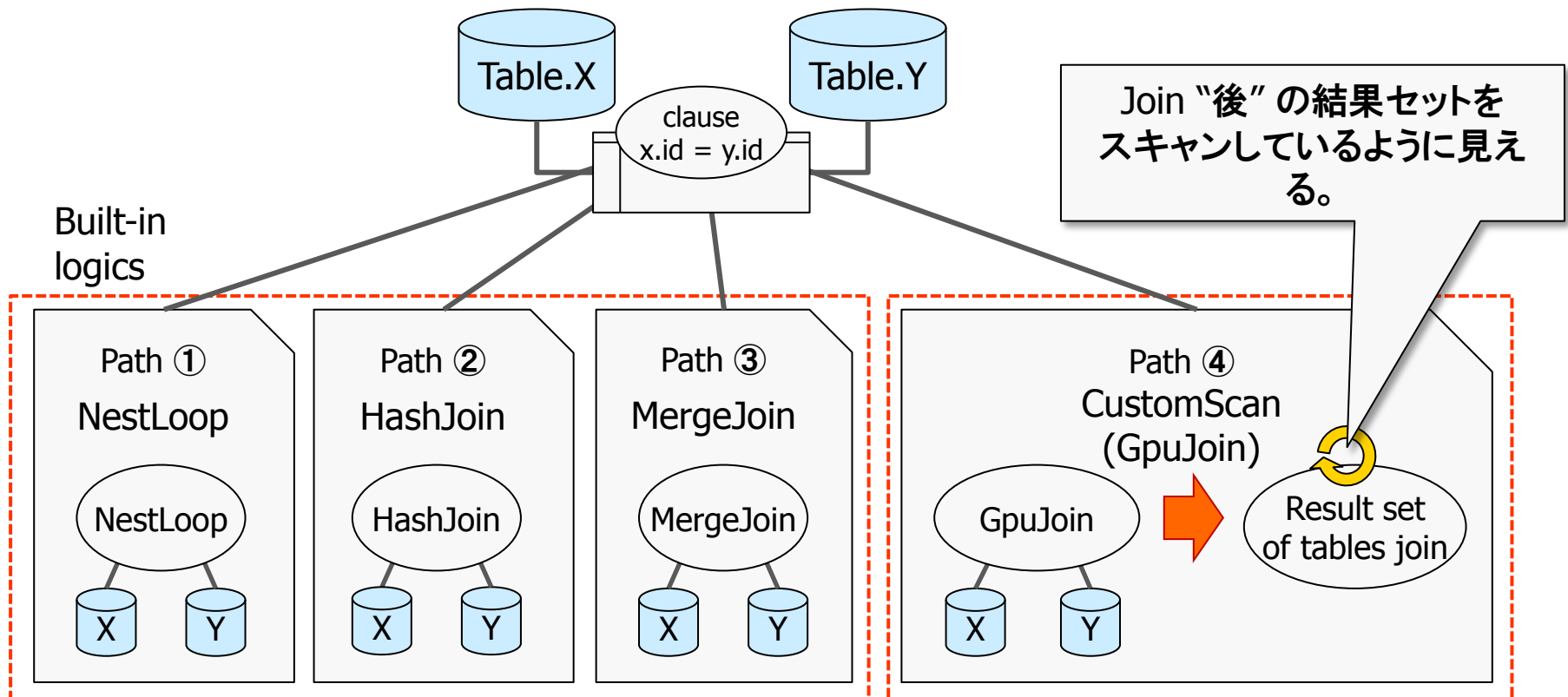
Executorコールバックを各自実装する

```
typedef struct CustomExecMethods
{
    const char *CustomName;
    /* Executor methods: mark/restore are optional, the rest are required */
    void          (*BeginCustomScan) (struct CustomScanState *node,
                                      Estate *estate,
                                      int eflags);
    TupleTableSlot (*ExecCustomScan) (struct CustomScanState *node);
    void          (*EndCustomScan) (struct CustomScanState *node);
    void          (*ReScanCustomScan) (struct CustomScanState *node);
    void          (*MarkPosCustomScan) (struct CustomScanState *node);
    void          (*RestrPosCustomScan) (struct CustomScanState *node);
    /* Optional: print additional information in EXPLAIN */
    void          (*ExplainCustomScan) (struct CustomScanState *node,
                                       List *ancestors,
                                       struct ExplainState *es);
} CustomExecMethods;
```



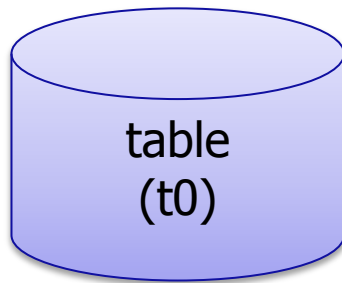
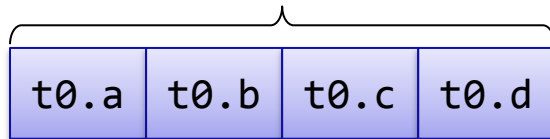
CustomJoinはどう見えるか？

CustomScanによる Join の置換え



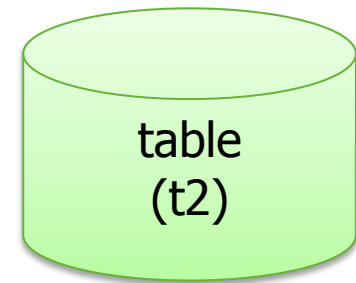
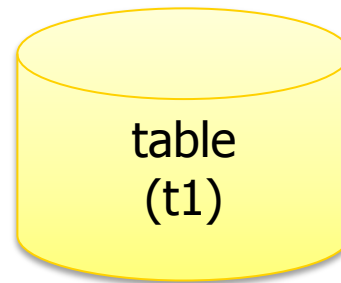
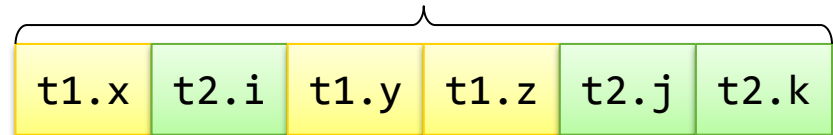
結果セットのレコード型

結果セットのレコード型は、
常にテーブル定義と一致



単純なスキャン

結果セットのレコード型は、
時と場合によって異なる

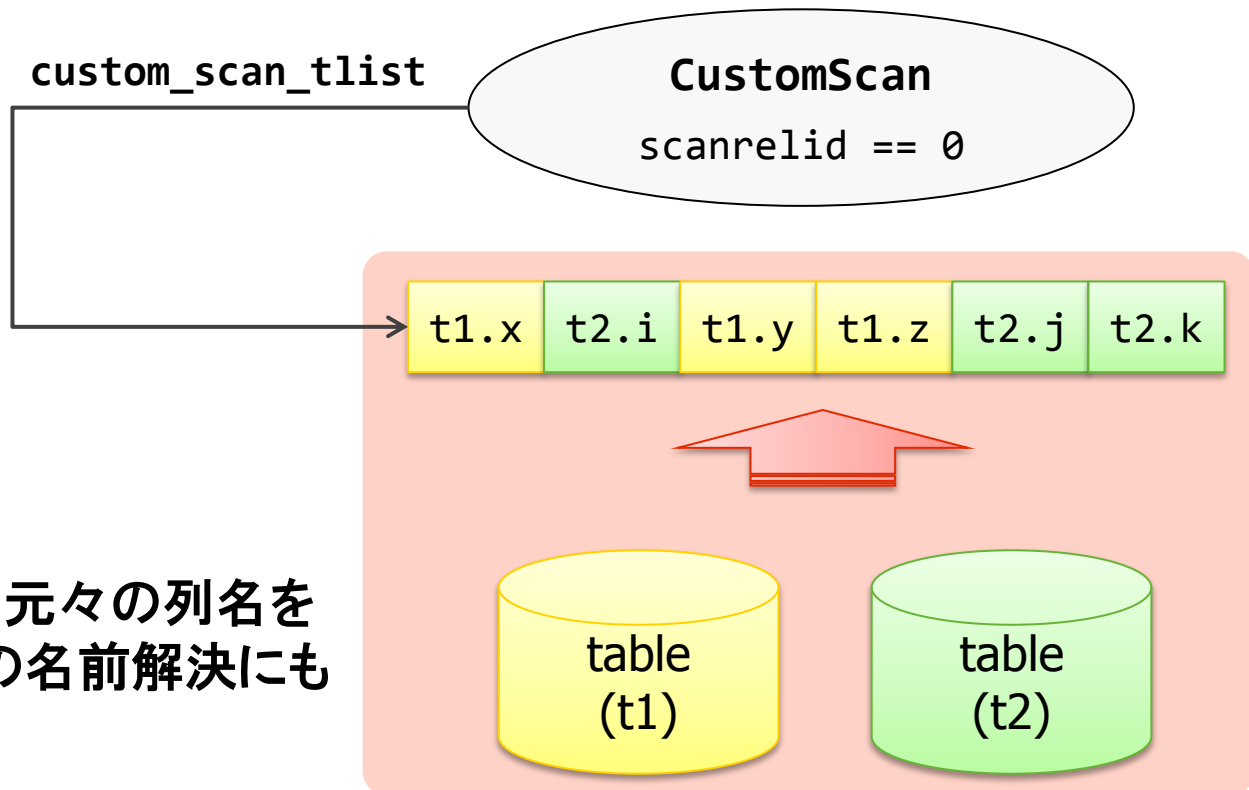


複数のテーブルから成るジョイン

scanrelid == 0 と custom_scan_tlist

scanrelid == 0

- CustomScanが特定の实テーブルスキャンでない事を示す Magic Number
- custom_scan_tlist で指定されたレコード型をスキャンすると見なす。



- EXPLAINで、元々の列名を表示する際の名前解決にも使用される。

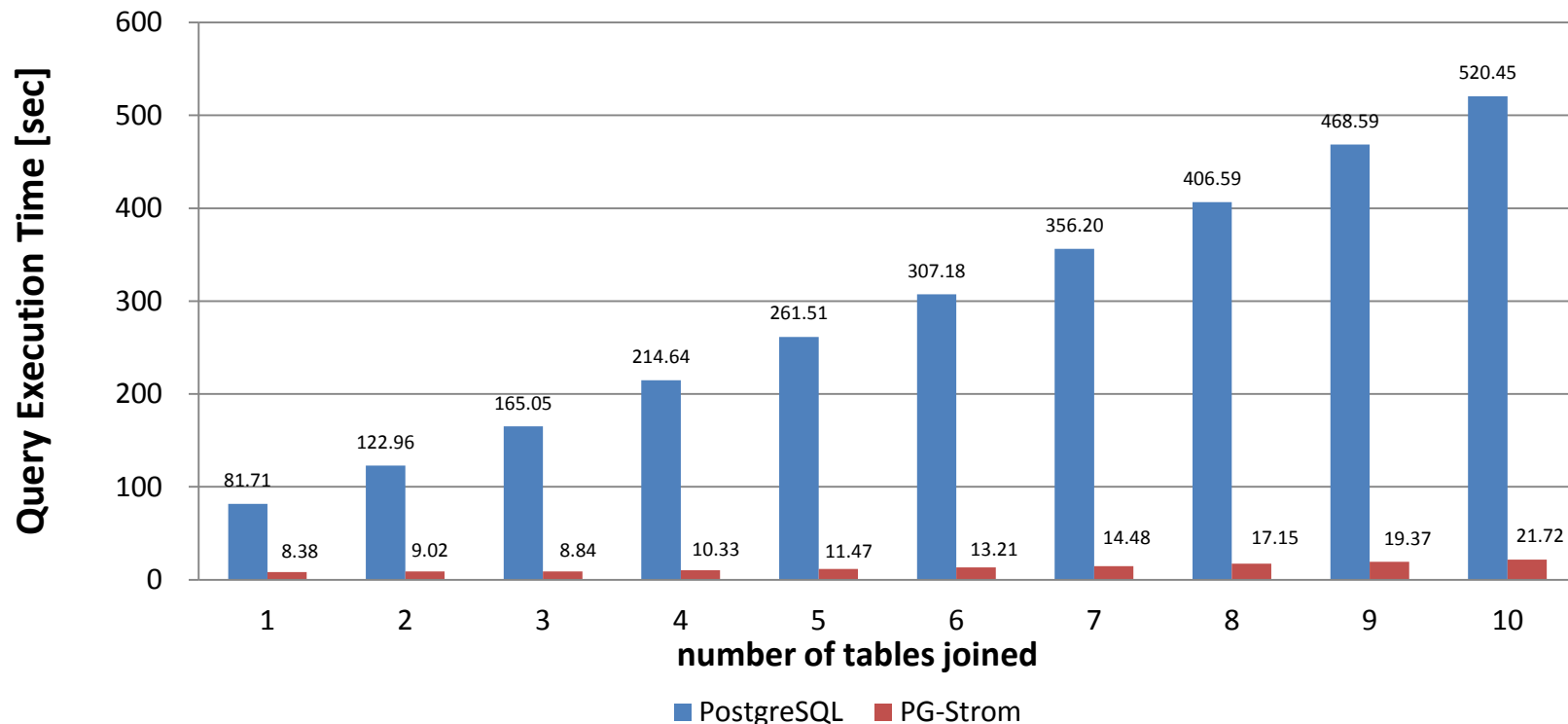
CustomScanを含むクエリ実行計画

```
postgres=# EXPLAIN SELECT cat, AVG(x)
FROM t0 NATURAL JOIN t1 RIGHT JOIN t2 ON t0.aid=t2.bid OR t0.bid=t2.bid
WHERE atext like '%abc%' GROUP BY cat;
QUERY PLAN
```

```
-----
HashAggregate (cost=348362.37..348362.70 rows=26 width=12)
  Group Key: t0.cat
    -> Custom Scan (GpuPreAgg) (cost=2134.00..348362.24 rows=26 width=52)
        Bulkload: On
        Reduction: Local + Global
        -> Custom Scan (GpuJoin) (cost=1134.00..347362.19 rows=1985 width=12)
            Bulkload: On
            Depth 1: Logic: GpuHashJoin, HashKeys: (aid), JoinQual: (aid = aid)
            Depth 2: Logic: GpuNestLoop, JoinQual: ((aid = bid) OR (bid = bid))
            -> Custom Scan (GpuScan) on t0 (cost=1000.00..143858.00 rows=10000060 ...)
            -> Custom Scan (MultiRels) (cost=834.05..343937.39 rows=4 width=4)
                Hash keys: aid
                nBatches: 1, Buckets: 1024, Buffer Usage: 2.33%
                -> Seq Scan on t1 (cost=0.00..834.00 rows=4 width=4)
                    Filter: (atext ~~ '%abc% '::text)
                -> Custom Scan (MultiRels) (cost=1134.00..347362.19 rows=40000 width=4)
                    nBatches: 1, Buffer Usage: 97.64%
                    -> Seq Scan on t2 (cost=0.00..734.00 rows=40000 width=4)
```



CustomにJoinを実装してみた結果



■ `SELECT cat, AVG(x) FROM t0 NATURAL JOIN t1 [, ...] GROUP BY cat;` をテーブル数を変えながら実行してクエリ応答時間を測定。

■ t0:1億行、t1~t10:それぞれ10万行を含む。全データはバッファにロード済み。

■ PostgreSQL v9.5devel + PG-Strom 5/26開発版、on CUDA7 (x86_64)

■ CPU: Xeon E5-2640, RAM: 256GB, GPU: NVIDIA GTX980, OS: RedHat EL7

v9.5に向けての Open Issue

Code review for foreign/cui X

GitHub, Inc. [US] <https://github.com/postgres/postgres/commit/1a8a4e5cde2b7755e11bde2ea7897bd650622>

This repository Search Explore Gist Blog Help kaigai +

postgres / postgres Unwatch 167 Star 1,418 Fork

Code review for foreign/custom join pushdown patch.

Commit e7cb7ee included some design decisions that seem pretty questionable to me, and there was quite a lot of stuff not to like about the documentation and comments. Clean up as follows:

- * Consider foreign joins only between foreign tables on the same server, rather than between any two foreign tables with the same underlying FDW handler function. In most if not all cases, the FDW would simply have had to apply the same-server restriction itself (far more expensively, both for lack of caching and because it would be repeated for each combination of input sub-joins), or else risk nasty bugs. Anyone who's really intent on doing something outside this restriction can always use the `set_join_pathlist_hook`.
- * Rename `fdw_ps_tlist/custom_ps_tlist` to `fdw_scan_tlist/custom_scan_tlist` to better reflect what they're for, and allow these custom scan tlists to be used even for base relations.
- * Change `make_foreignscan()` API to include passing the `fdw_scan_tlist` value, since the FDW is required to set that. Backwards compatibility doesn't seem like an adequate reason to expect FDWs to set it in some ad-hoc extra step, and anyway existing FDWs can just pass NIL.
- * Change the API of path-generating subroutines of `add_paths_to_joinrel`, and in particular that of `GetForeignJoinPaths` and `set_join_pathlist_hook`, so that various less-used parameters are passed in a struct rather than as separate parameter-list entries. The objective here is to reduce the probability that future additions to those parameter lists will result in source-level API breaks for users of these hooks. It's possible that this is even a small win for the core code, since most CPU architectures can't pass more than half a dozen parameters efficiently anyway. I kept `root`, `joinrel`, `outerrel`, `innerrel`, and `jointype` as separate parameters to reduce code churn in `joinpath.c` --- in particular, putting `jointype` into the struct would have been problematic because of the subroutines' habit of changing their local copies of that variable.

Browse files

拡張モジュールが
Child-Pathをプラン木に
変換できなくなってしまった!?

Code review for foreign/cui X

GitHub, Inc. [US] <https://github.com/postgres/postgres/commit/1a8a4e5cde2b7755e11bde2ea7897bd650622>

src/backend/optimizer/plan/createplan.c

```
83  @@ -44,6 +44,7 @@
44  44  #include "utils/lsyscache.h"
45  45
46  46
47  47  +static Plan *create_plan_recurse(PlannerInfo *root, Path *best_path);
48  48  static Plan *create_scan_plan(PlannerInfo *root, Path *best_path);
49  49  static List *build_path_tlist(PlannerInfo *root, Path *path);
50  50  static bool use_physical_tlist(PlannerInfo *root, RelOptInfo *rel);
51  51
52  52  @@ -219,7 +220,7 @@ create_foreignscan_plan(PlannerInfo *root, Path *best_path)
219  220  * create_plan_recurse
220  221  * Recursive guts of create_plan().
221  222  */
222  222  -Plan *
223  223  +static Plan *
224  224  create_plan_recurse(PlannerInfo *root, Path *best_path)
225  225  {
226  226  Plan *plan;
227  227
228  228  @@ -1950,7 +1951,7 @@ create_worktablescan_plan(PlannerInfo *root, Path *best_path,
1950  1951
1951  1952  /*
1952  1953  * create_foreignscan_plan
1953  1953  - * Returns a foreignscan plan for the base relation scanned by 'best_path'
1954  1954  + * Returns a foreignscan plan for the relation scanned by 'best_path'
1955  1955  * with restriction clauses 'scan_clauses' and targetlist 'tlist'.
```



もふっ、もふもふ。

