pg_hint_plan

NIPPON TELEGRAPH AND TELEPHONE CORPORATION

2025 年 09 月 29 日

目 次

第 1 章　概要 **3**

第 2 章　機能説明 **5**
2.1　基本的な使用方法 5

第 3 章　ヒントテーブル **7**
3.1　ヒントの種類 8
3.2　pg_hint_plan の GUC パラメータ 10

第 4 章　インストール **11**
4.1　Building binary module 11
4.2　Installing from a binary package 11
4.3　pg_hint_plan のロード 11

第 5 章　アンインストール **13**

第 6 章　ヒントの詳細 **15**
6.1　構文と配置 15
6.2　PL/pgSQL での使用 15
6.3　オブジェクト名の大文字と小文字の区別 16
6.4　オブジェクト名の特殊文字のエスケープ 16
6.5　複数出現するテーブルの区別 16
6.6　ビューまたはルールの根底にあるテーブル 17
6.7　継承 17
6.8　マルチステートメントでのヒント 18
6.9　VALUES 式 18
6.10　副問い合わせ 18
6.11　IndexOnlyScan ヒントの使用 19
6.12　NoIndexScan について 19
6.13　Parallel ヒントと UNION 19
6.14　Set ヒントによる pg_hint_plan のパラメータ設定 19
6.15　Using DisableIndex hint 20

第 7 章　エラー **21**
7.1　シンタックスエラー 21
7.2　誤ったオブジェクトの指定 21
7.3　冗長または競合するヒント 21
7.4　ネストされたコメント 21

第 8 章　機能的な制限事項 **23**
8.1　プランナ GUC パラメータの影響 23
8.2　実行不可能なプランの強制を試みるヒント 23

8.3 ECPG 内のクエリ 23
8.4 Query Identifiers 23

第 9 章 動作環境 **25**
9.1 関連項目 25
9.2 References 25

第 10 章 Hint list **27**

目次

- *pg_hint_plan 1.8*

第1章　概要

pg_hint_plan は SQL コメント内のヒント (/*+ SeqScan(a) */など) を用いることで実行計画を制御することができます。

PostgreSQL のプランナは静的なルールではなくデータの統計情報を用いたコストベースのオプティマイザを利用しています。プランナ（オプティマイザ）は SQL 文に対して可能な限りの実行計画のコストを推定し、最もコストが低い実行計画を選択します。プランナは最適な実行計画を選択するために最善を尽くしますが、データの特性やカラム間の相関を考慮していないため、必ずしも完璧ではありません。

第2章　機能説明

2.1 基本的な使用方法

pg_hint_plan は SQL 文に与えられた特別な形式のコメント内のヒント句を読み取ります。ヒントは先頭に "/*+"というシーケンスを付け、最後に "*/" を付けることで指定できます。ヒント句はヒント名とそれに続くパラメータを括弧で囲み、スペースで区切ったものです。ヒント句は読みやすくするために改行することができます。

以下の例では、pgbench_accounts に対してシーケンシャルスキャンを行う際に結合方法としてハッシュ結合が選択されています。

```
=# /*+
    HashJoin(a b)
    SeqScan(a)
   */
  EXPLAIN SELECT *
    FROM pgbench_branches b
    JOIN pgbench_accounts a ON b.bid = a.bid
    ORDER BY a.aid;
                                      QUERY PLAN
-------------------------------------------------------------------------------------
↪-
   Sort  (cost=31465.84..31715.84 rows=100000 width=197)
   Sort Key: a.aid
   ->  <b>Hash Join</b>  (cost=1.02..4016.02 rows=100000 width=197)
         Hash Cond: (a.bid = b.bid)
         ->  <b>Seq Scan on pgbench_accounts a</b>  (cost=0.00..2640.00␣
↪rows=100000 width=97)
         ->  Hash  (cost=1.01..1.01 rows=1 width=100)
             ->  Seq Scan on pgbench_branches b  (cost=0.00..1.01 rows=1 width=100)
(7 rows)
```

第3章　ヒントテーブル

ヒントは特別な形式のコメント内に記載されていますがクエリを編集できない場合には不便です。このような場合には "hint_plan.hints" という名前の特別なテーブルにヒントを置くことができます。このテーブルは以下のカラムで構成されています。

列名	説明
id	ユーザがヒントの行を識別するためのユニークな番号です。 この列はシーケンスによって自動的に埋められます。
query_id	A unique query ID, generated by the backend when the GUC compute_query_id is enabled
application_name	ヒント適用対象のアプリケーション名を指定します。下記の例では psql から実行されたクエリのみがヒントの適用対象となります。全てのアプリケーションにヒントを適用したいときは、空文字列を登録します。
hints	ヒント句を指定します。コメントの記号を除いたヒントのみを登録します。

以下の例はヒントテーブルの操作方法を示しています。

```
=# EXPLAIN (VERBOSE, COSTS false) SELECT * FROM t1 WHERE t1.id = 1;
                QUERY PLAN
----------------------------------------
 Seq Scan on public.t1
   Output: id, id2
   Filter: (t1.id = 1)
 Query Identifier: -7164653396197960701
(4 rows)
=# INSERT INTO hint_plan.hints(query_id, application_name, hints)
     VALUES (-7164653396197960701, '', 'SeqScan(t1)');
INSERT 0 1
=# UPDATE hint_plan.hints
     SET hints = 'IndexScan(t1)'
     WHERE id = 1;
UPDATE 1
=# DELETE FROM hint_plan.hints WHERE id = 1;
DELETE 1
```

ヒントテーブルは拡張機能の所有者が所有し、拡張機能作成時におけるデフォルトの権限を持ちます。ヒントテーブル内のヒントはコメント内のヒントよりも優先されます。

The query ID can be retrieved with pg_stat_statements or with EXPLAIN (VERBOSE).

3.1 ヒントの種類

Hinting phrases are classified in multiple types based on what kind of object and how they can affect the planner. See *Hint list* for more details.

3.1.1 スキャン方法

スキャン方法のヒントは、対象のテーブルに対して特定のスキャン方法を強制するものです。pg_hint_plan は対象のテーブルに別名が存在する場合、別名で認識します。この種類の例は SeqScan や IndexScan などです。

スキャン方法のヒントは、通常のテーブル・継承テーブル・UNLOGGED テーブル・一時テーブル・システムカタログに効果があります。外部テーブル・テーブル関数・VALUES 句・CTE・ビュー・副問い合わせには影響を与えません。

```
=# /*+
     SeqScan(t1)
     IndexScan(t2 t2_pkey)
    */
   SELECT * FROM table1 t1 JOIN table table2 t2 ON (t1.key = t2.key);
```

3.1.2 結合方法

結合方法のヒントは、指定したテーブルを含む結合の結合方法を強制するものです。

これは、通常のテーブル・継承テーブル・UNLOGGED テーブル・一時テーブル・外部テーブル・システムカタログ・テーブル関数・VALUES コマンド結果、およびパラメータリストに含めることが許可されている CTE の結合にのみ影響を与えます。しかし、ビュー・副問い合わせの結合には影響を与えません。

3.1.3 結合順

Leading ヒントは、2 つ以上のテーブルの結合順を強制するものです。強制には 2 つの方法があります。1 つは特定の結合順を強制し各結合レベルでは方向を制限しない方法です。もう 1 つは結合の方向を追加で指定するものです。詳細は**ヒント一覧**で確認してください。以下は例です。

```
=# /*+
     NestLoop(t1 t2)
     MergeJoin(t1 t2 t3)
     Leading(t1 t2 t3)
    */
   SELECT * FROM table1 t1
     JOIN table table2 t2 ON (t1.key = t2.key)
     JOIN table table3 t3 ON (t2.key = t3.key);
```

3.1.4 行数補正

Rows ヒントは、プランナの制限に起因する結合の見積り行数誤りを修正します。以下は例です。

```
=# /*+ Rows(a b #10) */ SELECT... ; Sets rows of join result to 10
=# /*+ Rows(a b +10) */ SELECT... ; Increments row number by 10
=# /*+ Rows(a b -10) */ SELECT... ; Subtracts 10 from the row number.
=# /*+ Rows(a b *10) */ SELECT... ; Makes the number 10 times larger.
```

3.1.5 パラレルプラン

Parallel ヒント は、スキャンの並列実行の設定を強制するものです。第 3 パラメータは強制の強さを指定します。soft は pg_hint_plan が max_parallel_worker_per_gather を変更するだけで、その他のすべてはプランナに任せることを意味します。hard はプランナのパラメータを変更し、強制的にその数値を適用するようにします。このヒントは通常のテーブル・継承の親テーブル・UNLOGGED テーブル・システムカタログに影響を与えることができます。外部テーブル・テーブル関数・VALUE 句・CTE・ビュー・サブクエリには影響を与えません。ビューの内部テーブルについては、対象オブジェクトとして実名/別名を用いて指定できます。次の例のクエリは、各テーブルで異なる設定を強制しています。

```
=# EXPLAIN /*+ Parallel(c1 3 hard) Parallel(c2 5 hard) */
   SELECT c2.a FROM c1 JOIN c2 ON (c1.a = c2.a);
                                QUERY PLAN
-------------------------------------------------------------------------------
 Hash Join  (cost=2.86..11406.38 rows=101 width=4)
   Hash Cond: (c1.a = c2.a)
   ->  Gather  (cost=0.00..7652.13 rows=1000101 width=4)
         Workers Planned: 3
         ->  Parallel Seq Scan on c1  (cost=0.00..7652.13 rows=322613 width=4)
   ->  Hash  (cost=1.59..1.59 rows=101 width=4)
         ->  Gather  (cost=0.00..1.59 rows=101 width=4)
               Workers Planned: 5
               ->  Parallel Seq Scan on c2  (cost=0.00..1.59 rows=59 width=4)

=# EXPLAIN /*+ Parallel(tl 5 hard) */ SELECT sum(a) FROM tl;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Finalize Aggregate  (cost=693.02..693.03 rows=1 width=8)
   ->  Gather  (cost=693.00..693.01 rows=5 width=8)
         Workers Planned: 5
         ->  Partial Aggregate  (cost=693.00..693.01 rows=1 width=8)
               ->  Parallel Seq Scan on tl  (cost=0.00..643.00 rows=20000 width=4)
```

3.1.6 プランニング中の GUC パラメータの設定

Set ヒントはプランニング中のみ GUC パラメータを変更します。 Query Planning で示した GUC パラメータは、他のヒントがプランナの設定パラメータと競合しない限り、期待される効果を発揮することができます。同じ GUC パラメータに関するヒントのうち、最後のものが効果を発揮します。*pg_hint_plan* **の** *GUC* **パラメータ** もこのヒントで設定可能ですが期待通りには動作しません。詳しくは**機能的な制限事項**を参照してください。

```
=# /*+ Set(random_page_cost 2.0) */
   SELECT * FROM table1 t1 WHERE key = 'value';
...
```

3.2 pg_hint_plan の GUC パラメータ

以下の GUC パラメータは pg_hint_plan の動作を制御します。

パラメータ名	説明	デフォルト値
pg_hint_plan.enable_hint	True は pg_hint_plan を有効にします。	on
pg_hint_plan.enable_hint_table	True はテーブルによってヒントを指定する機能を有効にします。	off
pg_hint_plan.parse_messages	指定したヒントを構文解析できなかった場合のログメッセージのレベルを指定します。指定可能な値は、error、warning、notice、info、log、debug です。	INFO
pg_hint_plan.debug_print	動作状況を示すログメッセージの出力を制御します。指定可能な値は off、on、detailed、verbose です。	off
pg_hint_plan.message_level	動作ログメッセージのログレベルを指定します。指定可能な値は、error、warning、notice、info、log、debug です。	INFO

第4章　インストール

このセクションは pg_hint_plan のインストール方法について説明します。

4.1 Building binary module

ソースツリーの先頭で make を実行し、適切なユーザで make install を実行してください。環境変数 PATH は、PostgreSQL のバイナリを指すように適切に設定する必要があります。

```
$ tar xzvf pg_hint_plan-1.x.x.tar.gz
$ cd pg_hint_plan-1.x.x
$ make
$ su
$ make install
```

4.2 Installing from a binary package

On Debian and Ubuntu pg_hint_plan is available as a binary package from the pgdg (PostgreSQL Global Development Group) repository. Assuming you've already added the repository to apt sources, installing the package is as simple as:

```
sudo apt install postgresql-<postgres version>-pg-hint-plan
```

Please visit https://www.postgresql.org/download/linux/ if you need help at adding the repository.

4.3 pg_hint_planのロード

pg_hint_plan は基本的に CREATE EXTENSION を必要としません。単純に LOAD コマンドで有効化できます。もちろん、postgresql.conf の shared_preload_libraries を設定することで全体にロード することもできます。また、特定のユーザに対し自動的にロードするための ALTER USER SET/ALTER DATABASE SET に興味があるかもしれません。

```
postgres=# LOAD 'pg_hint_plan';
LOAD
```

ヒントテーブルを使用する場合は、CREATE EXTENSION を実行し SET pg_hint_plan.enable_hint_table TO on を実行してください。

第5章　アンインストール

ソース ツリーからインストールし、それが利用可能なままになっている場合は、ソース ツリーの最上位ディレクトリにある make uninstall を実行するとインストールされたファイルがアンインストールされます。環境変数 PATH の設定が必要な場合があります。

```
$ cd pg_hint_plan-1.x.x
$ su
$ make uninstall
```

第6章　ヒントの詳細

6.1 構文と配置

pg_hint_plan は最初のブロックコメントのみからヒントを読み、アルファベット、数字、スペース、アンダースコア、カンマ、括弧の文字以外は構文解析を停止します。以下の例では、HashJoin(a b) と SeqScan(a) はヒントとして構文解析されますが、IndexScan(a) と MergeJoin(a b) はヒントとして構文解析されません。

```
=# /*+
     HashJoin(a b)
     SeqScan(a)
    */
   /*+ IndexScan(a) */
   EXPLAIN SELECT /*+ MergeJoin(a b) */ *
     FROM pgbench_branches b
     JOIN pgbench_accounts a ON b.bid = a.bid
     ORDER BY a.aid;
                                      QUERY PLAN
-------------------------------------------------------------------------------
↪-
 Sort  (cost=31465.84..31715.84 rows=100000 width=197)
   Sort Key: a.aid
   ->  Hash Join  (cost=1.02..4016.02 rows=100000 width=197)
         Hash Cond: (a.bid = b.bid)
         ->  Seq Scan on pgbench_accounts a  (cost=0.00..2640.00 rows=100000 width=97)
         ->  Hash  (cost=1.01..1.01 rows=1 width=100)
               ->  Seq Scan on pgbench_branches b  (cost=0.00..1.01 rows=1 width=100)
(7 rows)
```

6.2 PL/pgSQL での使用

pg_hint_plan は PL/pgSQL スクリプト内のクエリに対してはいくつかの制限付きで動作します。

- ヒントは以下のような種類のクエリにのみ影響します。
  - Queries that return one row (SELECT, INSERT, UPDATE and DELETE)
  - Queries that return multiple rows (RETURN QUERY)

  - ・動的な SQL 文 (EXECUTE)
  - ・カーソルを開く (OPEN)
  - ・クエリの結果をループ (FOR)

- ヒントコメントは次のようにクエリの最初の単語の後に配置する必要があります。クエリよりも先行するコメントはクエリの一部として送信されません。

```
=# CREATE FUNCTION hints_func(integer) RETURNS integer AS $$
   DECLARE
     id  integer;
     cnt integer;
   BEGIN
     SELECT /*+ NoIndexScan(a) */ aid
       INTO id FROM pgbench_accounts a WHERE aid = $1;
     SELECT /*+ SeqScan(a) */ count(*)
       INTO cnt FROM pgbench_accounts a;
     RETURN id + cnt;
   END;
   $$ LANGUAGE plpgsql;
```

6.3 オブジェクト名の大文字と小文字の区別

PostgreSQL がオブジェクト名を扱う方法とは異なり、pg_hint_plan はヒントに含まれるオブジェクト名とデータベース内部のオブジェクト名を大文字・小文字を区別して比較します。したがって、ヒント内のオブジェクト名 TBL はデータベース内の "TBL "にのみマッチし、TBL、tbl、Tbl のような引用符のない名前にはマッチしません。

6.4 オブジェクト名の特殊文字のエスケープ

The objects defined in a hint's parameter can use double quotes if they include parentheses, double quotes and white spaces. The escaping rules are the same as PostgreSQL.

6.5 複数出現するテーブルの区別

pg_hint_plan は別名が存在する場合、それを使用して対象オブジェクトを特定します。この動作は、1 つのテーブルが複数出現する中から特定のものを指定するのに便利です。

```
=# /*+ HashJoin(t1 t1) */
   EXPLAIN SELECT * FROM s1.t1
     JOIN public.t1 ON (s1.t1.id=public.t1.id);
INFO:  hint syntax error at or near "HashJoin(t1 t1)"
```

(次のページに続く)

(前のページからの続き)

```
DETAIL:  Relation name "t1" is ambiguous.
...
=# /*+ HashJoin(pt st) */
   EXPLAIN SELECT * FROM s1.t1 st
     JOIN public.t1 pt ON (st.id=pt.id);
                            QUERY PLAN
------------------------------------------------------------------------
 Hash Join  (cost=64.00..1112.00 rows=28800 width=8)
   Hash Cond: (st.id = pt.id)
   ->  Seq Scan on t1 st  (cost=0.00..34.00 rows=2400 width=4)
   ->  Hash  (cost=34.00..34.00 rows=2400 width=4)
         ->  Seq Scan on t1 pt  (cost=0.00..34.00 rows=2400 width=4)
```

6.6 ビューまたはルールの根底にあるテーブル

ヒントはビュー自体には適用されませんが、オブジェクト名がビュー上に展開されたクエリ内のオブジェクト名と一致する場合、ビュー内のクエリに影響を与えることができます。ビュー内のテーブルに別名を割り当てると、ビューの外からそれらを操作することができます。

```
=# CREATE VIEW v1 AS SELECT * FROM t2;
=# EXPLAIN /*+ HashJoin(t1 v1) */
          SELECT * FROM t1 JOIN v1 ON (c1.a = v1.a);
                            QUERY PLAN
------------------------------------------------------------------
 Hash Join  (cost=3.27..18181.67 rows=101 width=8)
   Hash Cond: (t1.a = t2.a)
   ->  Seq Scan on t1  (cost=0.00..14427.01 rows=1000101 width=4)
   ->  Hash  (cost=2.01..2.01 rows=101 width=4)
         ->  Seq Scan on t2  (cost=0.00..2.01 rows=101 width=4)
```

6.7 継承

Hints can only point to the parent of an inheritance tree and the hints affect all the tables in an inheritance tree. Hints pointing directly to inherited children have no effect.

6.8 マルチステートメントでのヒント

1 つのマルチステートメントに 1 つのヒントコメントを指定することができ、そのヒントはマルチステートメント内のすべてのステートメントに影響します。

6.9 VALUES 式

VALUES expressions in FROM clause are named as *VALUES* internally these can be hinted if it is the only VALUES of a query. Two or more VALUES expressions in a query cannot be distinguished by looking at an EXPLAIN result, resulting in ambiguous results:

```
=# /*+ MergeJoin(*VALUES*_1 *VALUES*) */
   EXPLAIN SELECT * FROM (VALUES (1, 1), (2, 2)) v (a, b)
     JOIN (VALUES (1, 5), (2, 8), (3, 4)) w (a, c) ON v.a = w.a;
INFO:  pg_hint_plan: hint syntax error at or near "MergeJoin(*VALUES*_1 *VALUES*) "
DETAIL:  Relation name "*VALUES*" is ambiguous.
                               QUERY PLAN
------------------------------------------------------------------------
 Hash Join  (cost=0.05..0.12 rows=2 width=16)
   Hash Cond: ("*VALUES*_1".column1 = "*VALUES*".column1)
   ->  Values Scan on "*VALUES*_1"  (cost=0.00..0.04 rows=3 width=8)
   ->  Hash  (cost=0.03..0.03 rows=2 width=8)
         ->  Values Scan on "*VALUES*"  (cost=0.00..0.03 rows=2 width=8)
```

6.10 副問い合わせ

サブクエリのコンテキストは ANY_subquery という名前を使用しヒントにすることができる場合があります。

```
IN (SELECT ... {LIMIT | OFFSET ...} ...)
= ANY (SELECT ... {LIMIT | OFFSET ...} ...)
= SOME (SELECT ... {LIMIT | OFFSET ...} ...)
```

これらの構文では副問い合わせを含むテーブルの結合を計画する際に、プランナは副問い合わせに対し内部的に名前を割り当てます。そのため、結合方法ヒントは暗黙の名前を使用している結合に適用することができます。以下は例です。

```
=# /*+HashJoin(a1 ANY_subquery)*/
   EXPLAIN SELECT *
     FROM pgbench_accounts a1
   WHERE aid IN (SELECT bid FROM pgbench_accounts a2 LIMIT 10);
                                        QUERY PLAN
```

(次のページに続く)

(前のページからの続き)

```
-------------------------------------------------------------------------------------
↪-------
 Hash Semi Join  (cost=0.49..2903.00 rows=1 width=97)
   Hash Cond: (a1.aid = a2.bid)
   ->  Seq Scan on pgbench_accounts a1  (cost=0.00..2640.00 rows=100000 width=97)
   ->  Hash  (cost=0.36..0.36 rows=10 width=4)
         ->  Limit  (cost=0.00..0.26 rows=10 width=4)
               ->  Seq Scan on pgbench_accounts a2  (cost=0.00..2640.00 rows=100000␣
↪width=4)
```

6.11 IndexOnlyScan ヒントの使用

IndexOnlyScan ヒントで指定されたインデックスでインデックスオンリースキャンを実行できない場合、インデックススキャンは予期せず別のインデックスで実行されることがあります。

6.12 NoIndexScan について

NoIndexScan ヒントは NoIndexOnlyScan を含んでいます。

6.13 Parallel ヒントと UNION

UNION が並列に実行できるのは、その下にあるすべてのサブクエリの並列実行が安全である場合のみです。したがってサブクエリのいずれかに並列実行を強制することで、並列実行可能な UNION が並列で実行されます。一方、ワーカーがゼロの PARALLEL ヒントはスキャンの並列実行を禁止します。

6.14 Set ヒントによる pg_hint_plan のパラメータ設定

pg_hint_plan parameters influence their own behavior so some parameters will not work as one could expect:

- ・enable_hint, enable_hint_table を変更するヒントは、デバッグログに"used hints"として報告されても無視されます。
- ・debug_print と message_level の設定は、対象クエリの処理の途中から動作します。

6.15 Using DisableIndex hint

A DisableIndex hint excludes the specified indexes from being considered during query planning. It takes precedence over other hints. A disabled index will not be used, even if explicitly requested by IndexScan.

```
=# /*+DisableIndex(t t_c1) IndexScan(t t_c1) */
   EXPLAIN SELECT * FROM t WHERE c1 = 1;
LOG:  indexes disabled for DisableIndex(t): t_c1
LOG:  available indexes for IndexScan(t):
LOG:  pg_hint_plan:
used hint:
DisableIndex(t t_c1)
not used hint:
IndexScan(t t_c1)
duplication hint:
error hint:

                            QUERY PLAN
------------------------------------------------------------------
 Index Scan using t_pkey on t  (cost=0.15..8.17 rows=1 width=12)
   Index Cond: (c1 = 1)
(2 rows)
```

第7章　エラー

pg_hint_plan はエラーが発生するとヒントの構文解析を停止し、既に構文解析されたヒントを使用します。以下は典型的なエラーです。

7.1 シンタックスエラー

構文的なエラーやヒント名の誤りなどはシンタックスエラーとして出力されます。これらのエラーは pg_hint_plan.debug_print が ON 以上の場合、pg_hint_plan.message_level によって指定されたメッセージレベルでサーバログに出力されます。

7.2 誤ったオブジェクトの指定

オブジェクトの指定に誤りがあるとヒントは無言で無視されます。この種類のエラーはシンタックスエラーと同様の条件で "Not used hints" としてサーバログに出力されます。

7.3 冗長または競合するヒント

冗長なヒントが指定されている場合やヒント同士が競合している場合は、最後のヒントが考慮されます。この種類のエラーは "duplication hints" として報告されます。

7.4 ネストされたコメント

ヒントコメントは再帰的に使用できません。 検出された場合、ヒントの構文解析は直ちに停止され、既に解析されたヒントはすべて無視されます。

第8章　機能的な制限事項

8.1 プランナ GUC パラメータの影響

プランナは、from_collapse_limit を超える FROM 句の項目に対する結合順を考慮しようとしません。pg_hint_plan は、このケースに対して期待される結合順に影響を与えることはできません。

8.2 実行不可能なプランの強制を試みるヒント

強制されたプランが実行できない場合、プランナは任意の実行可能なプランを選択します。

- ・FULL OUTER JOIN を nested loop で使用
- ・条件式で使用されるカラムを持っていないインデックスを使用
- ・ctid 条件が無いクエリに対する TID スキャンの実行

8.3 ECPG 内のクエリ

ECPG は埋め込み SQL として書かれたクエリのコメントを削除するのでヒントを渡すことはできません。唯一の例外は EXECUTE で、これはクエリ文字列をそのままサーバに渡します。このようなケースにおいてはヒントテーブルを利用することができます。

8.4 Query Identifiers

When compute_query_id is enabled, PostgreSQL generates a query ID, ignoring comments. Hence, queries with different hints, still written the same way, may compute the same query ID.

第9章　動作環境

pg_hint_plan 1.8 は PostgreSQL 18 のみをサポートします。

テスト済みの PostgreSQL バージョン

- ・バージョン 18

テスト済みの OS バージョン

- ・CentOS 8.5

9.1 関連項目

9.2 References

- EXPLAIN
- SET
- Server Config
- Parallel Plans

第10章　Hint list

使用可能なヒントは以下の通りです。

グループ	書式	説明
スキャン方法	SeqScan(テーブル)	指定されたテーブルにシーケンシャルスキャンを強制します。
	TidScan(テーブル)	指定されたテーブルに Tid スキャンを強制します。検索条件に ctid を指定した場合にのみ有効です。
	IndexScan(テーブル [インデックス...])	指定されたテーブルにインデックススキャンを強制します。指定されたインデックスがある場合は、そのインデックスに限定されます。
	IndexOnlyScan(テーブル [インデックス...])	指定されたテーブルにインデックスオンリースキャンを強制します。指定されたインデックスがある場合は、そのインデックスに限定されます。インデックスオンリースキャンが利用できない場合、インデックススキャンが使用されることがあります。
	BitmapScan(テーブル [インデックス...])	指定されたテーブルにビットマップスキャンを強制します。指定されたインデックスがある場合は、そのインデックスに限定されます。
	IndexScanRegexp(テーブル [POSIX 正規表現...])IndexOnlyScanRegexp(テーブル [POSIX 正規表現...])BitmapScanRegexp(テーブル [POSIX 正規表現...])	指定されたテーブルにインデックススキャン、インデックスオンリースキャン（PostgreSQL 9.2 以降)、ビットマップスキャンを強制します。指定された POSIX 正規表現パターンに一致するインデックスに限定されます。
	NoSeqScan(テーブル)	指定されたテーブルにシーケンシャルスキャンを行わないように強制します。
	NoTidScan(テーブル)	指定されたテーブルに TID スキャンを行わないように強制します。
	NoIndexScan(テーブル)	指定されたテーブルに対してインデックススキャン、インデックスオンリースキャンを行わないように強制します。
	NoIndexOnlyScan(テーブル)	指定されたテーブルに TID スキャンを行わないように強制します。
	NoBitmapScan(テーブル)	指定されたテーブルにビットマップスキャンを行わないように強制します。
Disable indexes	DisableIndex(table index...)	Disables the specified indexes during query planning, taking precedence over any other hints.
結合方法	NestLoop(テーブル テーブル [テーブル...])	指定されたテーブルで構成された結合にネステッドループ結合を強制します。
	HashJoin(テーブル テーブル [テーブル...])	指定されたテーブルで構成された結合にハッシュ結合を強制します。
	MergeJoin(テーブル テーブル [テーブル...])	指定されたテーブルで構成された結合にマージ結合を強制します。