PG Hint_Plan

Name

pg_hint_plan – controls execution plan with hinting phrases in comment of special form.

Synopsis

PostgreSQL uses cost based optimizer, which utilizes data statistics, not static rules. The planner (optimizer) esitimates costs of each possible execution plans for a SQL statement then the execution plan with the lowest cost finally be executed. The planner does its best to select the best best execution plan, but not perfect, since it doesn't count some properties of the data, for example, correlation between columns.

pg_hint_plan makes it possible to tweak execution plans using so-called "hints", which are simple descriptions in the SQL comment of special form.

Description

Basic Usage

pg_hint_plan reads hinting phrases in a comment of special form given with the target SQL statement. The special form is beginning by the character sequence "*/+" and ends with "*/". Hint phrases are consists of hint name and following parameters enclosed by parentheses and delimited by spaces. Each hinting phrases can be delimited by new lines for readability.

In the example below , hash join is selected as the joining method and scanning pgbench_accounts by sequential scan method.

The hint table

Hints are described in a comment in a special form in the above section. This is inconvenient in the case where queries cannot be edited. In the case hints can be placed in a special table named "hint_plan.hints". The table consists of the following columns.

column

description

id

Unique number to identify a row for a hint. This column is filled automatically by sequence.

norm_query_string

A pattern matches to the query to be hinted. Constants in the query have to be replace with '?' as in the following example. White space is significant in the pattern.

application_name

The value of application_name of sessions to apply the hint. The hint in the example below applies to sessions connected from psql. An empty string means sessions of any application_name.

hints

Hint phrase. This must be a series of hints excluding surrounding comment marks.

The following example shows how to operate with the hint table.

The hint table is owned by the creator user and having the default previledges at the time of creation. during CREATE EXTENSION. Table hints are prioritized than comment hits.

The types of hints

Hinting phrases are classified into six types based on what kind of object and how they can affect planning. Scaning methods, join methods, joining order, row number correction, parallel query and GUC setting. You will see the lists of hint phrases of each type in Hint list.

Hints for scan methods

Scan method hints enforce specific scanning method on the target table. pg_hint_plan recognizes the target table by alias names if any. They are 'SeqScan' , 'IndexScan' and so on in this kind of hint.

Scan hints are effective on ordinary tables, inheritance tables, UNLOGGED tables, temporary tables and system catalogs. External(foreign) tables, table functions, VALUES clause, CTEs, views and subquiries are not affected.

Hints for join methods

Join method hints enforce the join methods of the joins involving specified tables.

This can affect on joins only on ordinary tables, inheritance tables, UNLOGGED tables, temporary tables, external (foreign) tables, system catalogs, table functions, VALUES command results and CTEs are allowed to be in the parameter list. But joins on views and sub query are not affected.

Hint for joining order

This hint "Leading" enforces the order of join on two or more tables. There are two ways of enforcing. One is enforcing specific order of joining but not restricting direction at each join level. Another enfoces join direction additionaly. Details are seen in the hint list table.

Hint for row number correction

This hint "Rows" corrects row number misestimation of joins that comes from restrictions of the planner.

Hint for parallel plan

This hint "Parallel" enforces parallel execution configuration on scans. The third parameter specifies the strength of enfocement. "soft" means that pg_hint_plan only changes max_parallel_worker_per_gather and leave all others to planner. "hard" changes other planner parameters so as to forcibly apply the number. This can affect on ordinary tables, inheritnce parents, unlogged tables and system catalogues. External tables, table functions, values clause, CTEs, views and subqueries are not affected. Internal tables of a view can be specified by its real name/alias as the target object. The following example shows that the query is enforced differently on each table.

GUC parameters temporarily setting

'Set' hint changes GUC parameters just while planning. GUC parameter shown in Query Planning can have the expected effects on planning unless any other hint conflicts with the planner method configuration parameters. The last one among hints on the same GUC parameter makes effect. GUC parameters for pg_hint_plan are also settable by this hint but it won't work as your expectation. See Restrictions for details.

GUC parameters for pg_hint_plan

GUC parameters below affect the behavior of pg_hint_planpg_hint_plan.

Parameter name

discription

Default

pg_hint_plan.enable_hint

True enbles pg_hint_plan.

on

pg_hint_plan.enable_hint_table

True enbles hinting by table. true or false.

off

pg_hint_plan.parse_messages

Specifies the log level of hint parse error. Valid values are error, warning, notice, info, log, debug.

INFO

pg_hint_plan.debug_print

Controls debug print and verbosity. Valid vaiues are off, on, detailed and verbose.

off

pg_hint_plan.message_level

Specifies message level of debug print. Valid values are error, warning, notice, info, log, debug.

LOG

Installation

This section describes the installation steps.

building binary module

Simply run "make" in the top of the source tree, then "make install" as appropriate user. The PATH environment variable should be set properly for the target PostgreSQL for this process.

Loding pg_hint_plan

Basically pg_hint_plan does not requires CREATE EXTENSION. Simply loading it by LOAD command will activate it and of course you can load it globally by setting shared_preload_libraries in postgresql.conf. Or you might be interested in ALTER USER SET/ALTER DATABASE SET for automatic loading for specific sessions.

Do CREATE EXTENSION and SET pg_hint_plan.enable_hint_tables TO on if you are planning to hint tables.

Unistallation

"make uninstall" in the top directory of source tree will uninstall the installed files if you installed from the source tree and it is left available.

Details in hinting

Syntax and placement

pg_hint_plan reads hints from only the first block comment and any characters except alphabets, digits, spaces, underscores, commas and parentheses stops parsing immediately. In the following example HashJoin(a b) and SeqScan(a) are parsed as hints but IndexScan(a) and MergeJoin(a b) are not.

Using with PL/pgSQL

pg_hint_plan works for queries in PL/pgSQL scripts with some restrictions.

Hints affect only on the following kind of queires.

Queries that returns one row. (SELECT, INSERT, UPDATE and DELETE)

Queries that returns multiple rows. (RETURN QUERY)

Dynamic SQL statements. (EXECUTE)

Cursor open. (OPEN)

Loop over result of a query (FOR)

A hint comment have to be placed after the first word in a query as the following since preceding comments are not sent as a part of the query.

Letter case in the object names

Unlike the way PostgreSQL handles object names, pg_hint_plan compares bare object names in hints against the database internal object names in case sensitive way. Therefore an object name TBL in a hint matches only "TBL" in database and does not match any unquoted names like TBL, tbl or Tbl.

Escaping special chacaters in object names

The objects as the hint parameter should be enclosed by double quotes if they includes parentheses, double quotes and white spaces. The escaping rule is the same as PostgreSQL.

Distinction between multiple occurances of a table

pg_hint_plan identifies the target object by using aliases if exists. This behavior is usable to point a specific occurance among multiple occurances of one table.

Underlying tables of views or rules

Hints are not applicable on views itself, but they can affect the queries within if the object names match the object names in the expanded query on the view. Assigning aliases to the tables in a view enables them to be manipulated from outside the view.

Inheritance tables

Hints can point only the parent of an inheritance tables and the hint affect all the inheritance. Hints simultaneously point directly to children are not in effect.

Hinting on multistatements

One multistatement can have exactly one hint comment and the hints affects all of the individual statement in the multistatement. Notice that the seemingly multistatement on the interactive interface of psql is internally a sequence of single statements so hints affects only on the statement just following.

VALUES expressions

VALUES expressions in FROM clause are named as *VALUES* internally so it is hintable if it is the only VALUES in a query. Two or more VALUES expressions in a query seems distinguishable looking its explain result. But in reality it is mere a cosmetic and they are not distinguisable.

Subqueries

Subqueries in the following context occasionally can be hinted using the name "ANY_subquery".

For these syntaxes, planner internally assigns the name to the subquery when planning joins on tables including it, so join hints are applicable on such joins using the implicit name as the following.

Using IndexOnlyScan hint

Index scan may unexpectedly performed on another index when the index specifed in IndexOnlyScan hint cannot perform index only scan.

Behavior of NoIndexScan

NoIndexScan hint involes NoIndexOnlyScan.

Parallel hint and UNION

A UNION can run in parallel only when all underlying subqueries are parallel-safe. Conversely enforcing parallel on any of the subqueries let a parallel-executable UNION run in parallel. Meanwhile, a parallel hint with zero workers hinhibits a scan from executed in parallel.

Setting pg_hint_plan parameters by Set hints

pg_hint_plan paramters change the behavior of itself so some parameters doesn't work as expected.

Hints to change enable_hint, enable_hint_tables are ignored even though they are reported as “used hints” in debug logs.

Setting debug_print and message_level works from midst of the processing of the target query.

Errors

pg_hint_plan stops parsing on any error and uses hints already parsed on the most cases. Followings are the typical errors.

Syntax errors

Any syntactical errors or wrong hint names are reported as an syntax error. These errors are reported in the server log with the message level which specified by pg_hint_plan.message_level if pg_hint_plan.debug_print is on and aboves.

Object misspecifications

Object misspecifications results silent ingorance of the hints. This kind of error is reported as “not used hints” in the server log by the same condtion to syntax errors.

Redundant or conflicting hints

The last hint will be active when redundant hints or hints conflicting with each other. This kind of error is reported as “duplication hints” in the server log by the same condition to syntax errors.

Nested comments

Hint comment cannot include another block comment within. If pg_hint_plan finds it, differently from other erros, it stops parsing and abandans all hints already parsed. This kind of error is reported in the same manner as other errors.

Functional limitations

Influences of some of planner GUC parameters

The planner does not try to consider joining order for FROM clause entries more than from_collapse_limit. pg_hint_plan cannot affect joining order as expected for the case.

Hints trying to enforce unexecutable plans

Planner chooses any executable plans when the enforced plan cannot be executed.

FULL OUTER JOIN to use nested loop

To use indexes that does not have columns used in quals

To do TID scans for queries without ctid conditions

Queries in ECPG

ECPG removes comments in queries written as embedded SQLs so hints cannot be passed form those queries. The only exception is that EXECUTE command passes given string unmodifed. Please consider hint tables in the case.

Work with pg_stat_statements

pg_stat_statements generates a query id ignoring comments. As the result the identical queires with different hints are summerized as the same query.

Requirements

pg_hint_plan13 1.3 requires PostgreSQL 13.

PostgreSQL versions tested

Version 13

OS versions tested

CentOS 8.2

See also

PostgreSQL documents

EXPLAIN SET Server Config Parallel Plans

pg_hint_plan

Appendix A. Hints list

pg_hint_plan 1.3 Appendices

pg_hint_plan > Appendix A. Hints list

Appendix A. Hints list

The available hints are listed below.

Group

Format

Description

Scan method

SeqScan(table)

Forces sequential scan on the table

TidScan(table)

Forces TID scan on the table.

IndexScan(table[index...])

Forces index scan on the table. Restricts to specified indexes if any.

IndexOnlyScan(table[index...])

Forces index only scan on the table. Rstricts to specfied indexes if any. Index scan may be used if index only scan is not available. Available for PostgreSQL 9.2 and later.

BitmapScan(table[index...])

Forces bitmap scan on the table. Restoricts to specfied indexes if any.

IndexScanRegexp(table[POSIX Regexp...])IndexOnlyScanRegexp(table[POSIX Regexp...])BitmapScanRegexp(table[POSIX Regexp...])

Forces index scan or index only scan (For PostgreSQL 9.2 and later) or bitmap scan on the table. Restricts to indexes that matches the specified POSIX regular expression pattern

NoSeqScan(table)

Forces not to do sequential scan on the table.

NoTidScan(table)

Forces not to do TID scan on the table.

NoIndexScan(table)

Forces not to do index scan and index only scan (For PostgreSQL 9.2 and later) on the table.

NoIndexOnlyScan(table)

Forces not to do index only scan on the table. Available for PostgreSQL 9.2 and later.

NoBitmapScan(table)

Forces not to do bitmap scan on the table.

Join method

NestLoop(table table[table...])

Forces nested loop for the joins consist of the specifiled tables.

HashJoin(table table[table...])

Forces hash join for the joins consist of the specifiled tables.

MergeJoin(table table[table...])

Forces merge join for the joins consist of the specifiled tables.

NoNestLoop(table table[table...])

Forces not to do nested loop for the joins consist of the specifiled tables.

NoHashJoin(table table[table...])

Forces not to do hash join for the joins consist of the specifiled tables.

NoMergeJoin(table table[table...])

Forces not to do merge join for the joins consist of the specifiled tables.

Join order

Leading(table table[table...])

Forces join order as specified.

Leading(<join pair>)

<td>Forces join order and directions as specified. A join pair is a pair of tables and/or ot

Row number correction

<td nowrap>Rows(table table[table...] correction)</td>

Corrects row number of a result of the joins consist of the specfied tables. The available correction methods are absolute (#&ltn&gt), addition (+&ltn&gt), subtract (-&ltn&gt) and multiplication (*&ltn&gt). &ltn&gt should be a string that strtod() can read.

Parallel query configuration

<td nowrap>Parallel(table &lt# of workers&gt [soft|hard])</td>

Enforce or inhibit parallel execution of specfied table. &lt# of workers&gt is the desired number of parallel workers, where zero means inhibiting parallel execution. If the third parameter is soft (default), it just changes max_parallel_workers_per_gather and leave everything else to planner. Hard means enforcing the specified number of workers.

GUC

Set(GUC-param value)

Set the GUC parameter to the value while planner is running.

pg_hint_plan > Appendix A. hints list