云数据库 GaussDB

向量数据库开发指南（集中式_V2.0-10.x）

文档版本 01

发布日期 2026-07-20

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 产品描述

背景

在大数据时代的浪潮中，数据量爆炸式增长，数据类型复杂化，特别是随着机器学习和人工智能的兴起，对于能够有效处理高维度、非结构化数据的需求日益迫切。这些非结构化数据（如图像、音频、视频和复杂文本等），通常以高维向量的形式表示，其处理和检索挑战着传统关系型数据库的极限。在这样的背景下，向量数据库应运而生，旨在满足这些新兴需求，提供更高效的数据处理和检索解决方案。

向量数据库概念

向量数据库是一种专门设计用于存储、管理和检索高维度向量数据的数据库系统。与传统关系型数据库相比，它们在数据模型、索引机制和查询处理上有显著的不同。向量数据库通过优化的索引技术（如近似最近邻搜索）来提高非结构化数据的检索效率，同时，它们也能够支持复杂的数据分析和机器学习模型，使得数据处理更加高效和智能。

应用场景

在大数据和机器学习模型迅速发展的当下，向量数据库在多个领域展现出其独特价值。从推荐系统到图像识别，从自然语言处理到复杂数据分析，向量数据库的高效检索能力和灵活的数据处理机制使其成为这些应用的理想选择。在这些场景中，向量数据库不仅加快了数据处理速度，还提高了分析的准确性和深度，为数据驱动的决策提供了强大的支持。

向量数据库的主要应用场景包括：

- **推荐系统**：例如，在视频推荐、新闻推荐、购物推荐等场景。
- **图像识别**：例如，通过图片搜索商品、人脸识别等场景。
- **非结构化数据处理**：例如，音频、视频、图像等数据可以通过转化处理得到相应的向量数据，再进行关键性分析。
- **大模型智能问答系统：**例如，向量数据库被用于改进语义搜索引擎，通过存储和检索文本内容的向量表达，提供更准确的搜索结果。

工作原理

向量数据库将当前主流的基于磁盘的向量检索算法融合进GaussDB数据库中，让数据库用户能够使用原生SQL进行向量类型数据的创建、导入和索引构建，查询计划生成

以及高效向量检索；在当前版本中，用户可以将向量类型（FloatVector和BoolVector）当做原生类型一样使用，可以通过创建GsIVFFLAT索引或者GsDiskANN索引加速TopK ANN查询。GsIVFFLAT索引通过利用聚类算法将高维向量空间按照距离划分成不同的桶，然后向量检索过程中可以通过查询向量和向量分桶中心点的距离筛选出候选的向量检索集，因此检索代价相比于全量扫描会显著降低，如下图1-1所示。GsDiskANN索引的原理是对所有的向量点寻找最近邻向量，构建一个稀疏图结构，并且利用松弛系数产生一些捷径边（Shortcut Edge）加速查询，如下图1-2所示。在此基础上，GaussDB优化了数据删除逻辑，保证性能的基础上能够做到实时删除感知，并且长时间运行不损失精度。GaussDB向量索引支持高并发检索和修改，底层存储支持Astore和Ustore两种，支持MVCC并发控制。GaussDB向量数据库增加了日志类型，支持完整的高可用能力，比如集中式主备同步、并行回放、极致RTO等能力。

图 1-1 GsIVFFLAT 原理示意图

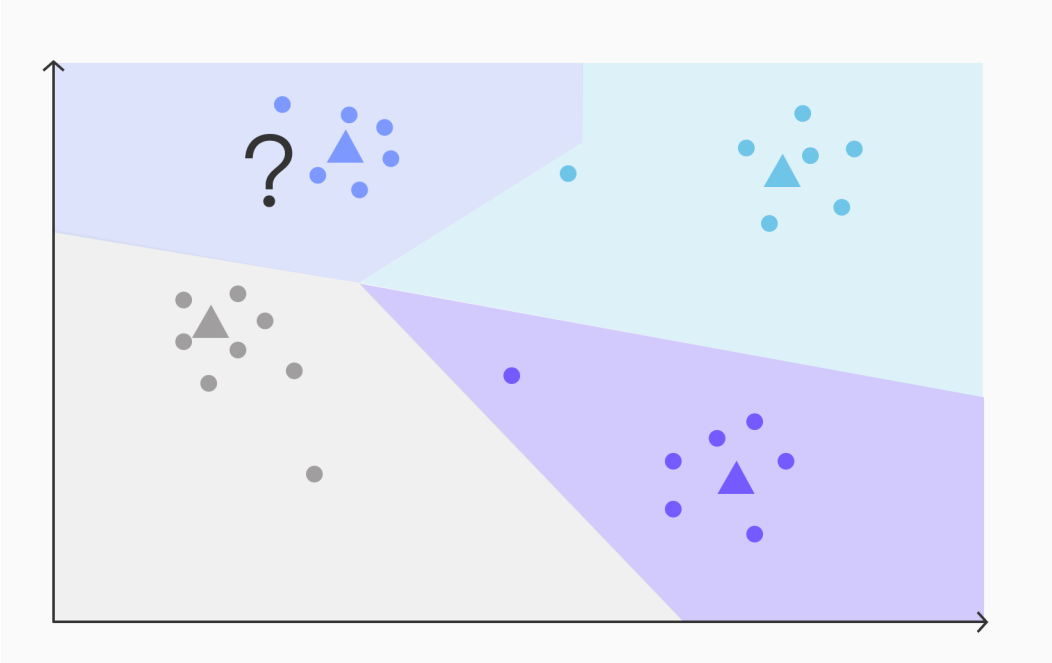
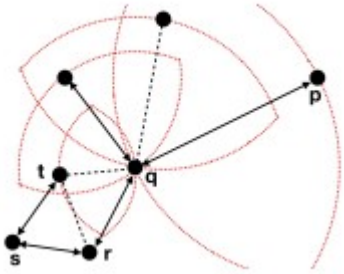
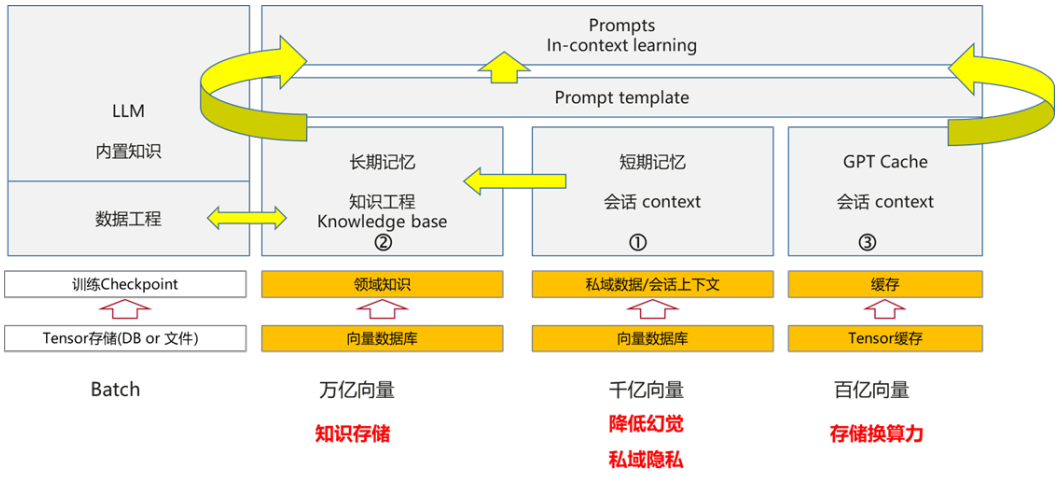


图 1-2 GsDiskANN 原理示意图



在大模型场景中，向量数据库作为大模型技术栈中的“海马体”，承担着长短期记忆的职责。具体来说分为三种能力，长期记忆、短期记忆和临时缓存，如图1-3所示。

图 1-3 向量数据库在大模型时代的角色



- 长期记忆：主要用于知识库场景。业务特点是万亿级别数据量，对于检索召回率要求很高，向量数据库可以通过相似检索对Prompt做补充，提高回答准确率和效率，解决大模型“幻觉”问题。
- 短期记忆：主要用于会话上下文场景。业务特点是千亿级别数据量，要求高效的数据插入和检索，向量数据库可以通过相似检索补充新问题Prompt，生成新问题答案并且保存在数据库中。
- 临时缓存：主要用于答案生成过程加速场景。业务特点是数据量大，要求高效的数据更新和检索，向量数据库临时保存之前出现过的字符前缀embedding以及需要产生的下一个字符，通过检索可以加速答案生成过程。

使用约束

1. 创建表维度不能超过4096，即维度范围在[1, 4096]。本版本向量索引，仅GsDiskANN支持4096维，其它类型索引支持到1024维。
2. 不支持创建分区表全局索引（gpi）。
3. 向量索引不支持创建Unique索引。
4. 向量索引不支持聚簇排序。
5. GsDiskANN支持创建向量标量混合多列索引，其他向量索引不支持创建多列索引。
6. floatvector和boolvector不支持作为分区键。
7. floatvector和boolvector不支持创建主键。
8. floatvector和boolvector不支持创建Unique键。
9. floatvector和boolvector不支持为NULL值。
10. floatvector和boolvector作为创建函数的参数和返回值时，无法进行维度校验。
11. dblink暂不支持vector类型，因此也不支持向量索引创建。
12. 本版本向量索引，GsIVFFLAT和GsDiskANN支持备机读，BM25和向量标量混合索引不支持备机读。
13. 向量索引GsDiskANN支持增量构建，GsIVFFLAT索引和GsDiskANN索引支持在线创建（不支持二级分区表），BM25不支持在线创建。
14. floatvector和boolvector类型不支持创建除向量索引以外的索引(btree，ubtree)。

15. 当前向量索引只支持距离TopK语句（按照距离升序排序），其他距离操作不进行索引。
16. 索引构建时间较长，内存规格低的场景需要预留构建时间。
17. 图索引空间膨胀率较大，磁盘规格低的场景无法创建成功。
18. 向量检索是资源密集型操作，如果同时进行多个业务，会影响性能。
19. 不支持unlogged、temp表构建向量索引。
20. 使用ustore索引，相比于astore索引空间增长10%以内，检索时延增加5%。
21. 索引依赖Vacuum进行空间回收，两次Vacuum之间如果存在频繁增删，向量检索会出现效率下降、空间膨胀的问题。
22. 索引类型不支持统计信息，选择率估计使用默认值，有代价估计，但是目前只支持索引和顺序扫描进行选择。
23. 本版本向量索引，GsDiskANN, BM25不支持SMP并行扫描，GsIVFFLAT支持SMP并行扫描。
24. AI WatchDog 特性与向量特性互斥，建议禁用AI WatchDog与向量交互的场景。

须知

当前版本使用分区表和分区索引时，避免分区DDL操作和二级分区索引的使用（包括分区合并，分区拆分，自动扩展分区等），可能会出现一些索引的错误。

2 使用向量数据库

2.1 使用入门

在机器学习和自然语言处理中，Embedding是指将高维度的数据（例如文字、图片、音频）映射到低维度空间的过程。Embedding向量通常是一个由实数构成的向量，它将输入的数据表示成一个连续的数值空间中的点。简而言之，Embedding是一个N维的实值向量，可以用来表示任何事情，如文本、音乐、视频等。

本小节介绍向量数据库的基本功能，包括创建向量表、修改向量属性类型、管理向量数据、清空表格等操作。

步骤1 创建含向量类型的表，同时设定数据维度。建表时向量类型必须要指定维度，当插入数据的维度与设置的数据维度不一致时，数据库报错。

```
gaussdb=# CREATE TABLE t1(id int unique, repr floatvector(4));
gaussdb=# CREATE TABLE t2(id int unique, repr boolvector(3));
```

步骤2 增加向量列。需指定默认值，否则数据库报错。

```
gaussdb=# ALTER TABLE t1 ADD COLUMN floatvec floatvector(3) default '[1,1,1]';
```

步骤3 修改表中的向量数据类型属性。可以通过ALTER TABLE语句修改向量维度。

```
gaussdb=# ALTER TABLE t1 ALTER COLUMN repr type floatvector(4);
```

注意：如果表中已经插入数据，变换向量维度时需要表中数据符合维度设定，否则数据库报错；当向量属性已创建向量索引，属性的数据类型无法切换到其他数据类型，数据库报错。

步骤4 插入数据。

```
gaussdb=# INSERT INTO t1 VALUES(0, '[30,12,12,25]');
gaussdb=# INSERT INTO t1 VALUES(1, '{7.5,62,1,235}');
```

字符型数据隐式转化为向量数据类型。

```
gaussdb=# INSERT INTO t2 VALUES(2, '[1, 0, 1]');
gaussdb=# INSERT INTO t2 VALUES(3, '[T, T, F]');
gaussdb=# INSERT INTO t2 VALUES(4, '[false, true, true]');
gaussdb=# INSERT INTO t2 VALUES(5, '[Y, N, N]');
gaussdb=# INSERT INTO t2 VALUES(6, '[no, yes, no]');
gaussdb=# INSERT INTO t2 VALUES(7, '[off, off, on]');
gaussdb=# INSERT INTO t2 VALUES(8, '[1, yes, on]');
```

步骤5 删除数据。


```
gaussdb=# DELETE t1 where repr <-> '[30,12,12,25]' < 0.2;
```

步骤6 对向量列和非向量列的更新和删除操作，以及VACUUM操作。

```
gaussdb=# UPDATE t1 SET id = 38;
gaussdb=# UPDATE t1 SET repr = '[1000,152,132,5]';
gaussdb=# DELETE FROM t1 WHERE id=38;
gaussdb=# DELETE FROM t1 WHERE repr='[30,12,12,25]';
gaussdb=# DELETE FROM t1 WHERE repr <-> '[30,12,12,25]' < 0.8;
gaussdb=# VACUUM t1;
```

----结束

2.2 数据类型转换

向量数据库支持向量数据类型之间的转换，数据类型转换中向量数据类型可以和相对应的数组类型进行自由显式转换：array<->vector。

- array -> vector

  使用函数，显式转换。

  ```
  gaussdb=# SELECT floatvector(ARRAY[1,2,9.3]);
  ```

- vector -> array

  使用函数，显式转换。

  ```
  gaussdb=# SELECT vector_to_array(floatvector('[1,2,9.3]'));
  ```

- string -> vector

  显式转换。

  ```
  gaussdb=# CREATE TABLE t1 (id int, b varchar);
  gaussdb=# INSERT INTO t1 VALUES (1, '[2.3, 1.8, 3.6]');
  gaussdb=# SELECT '[1,2,3]' <-> b::floatvector from t1;
  gaussdb=# SELECT '[1,2,3]' <-> CAST(b AS floatvector) from t1;
  ```

  隐式转换。

  ```
  gaussdb=# CREATE TABLE t2 (b floatvector(3));
  gaussdb=# INSERT INTO t2 VALUES ('[2.3, 1.8, 3.6]');
  gaussdb=# SELECT '[1,2,3]' <-> b from t2;
  ```

表 2-1 字符类型、向量类型转换映射关系

转换方式	支持转换类型
CAST(::)	• char(n) • character(n) • nchar(n) • varchar(n) • character varying(n) • varchar2(n) • nvarchar2 • clob • text • bpchar

转换方式	支持转换类型
隐式转换	unknown（如果没有为字符串文本声明类型，则该文本首先被定义成一个unknown类型。参见《参考》中“SQL参考 > 类型转换”章节内容。）

2.3 运算符计算

向量数据库支持二目运算符，针对向量数据进行简单的加减运算、内积运算和大小比较。

- 加减（+、-）
  ```
  gaussdb=# SELECT floatvector('[1,2,3]') - floatvector('[1,2,3]');
  ```
  说明：仅floatvector支持向量加减操作。
- 内积，向量数据类型不支持*运算符，可使用系统函数inner_product，进行计算。
  ```
  gaussdb=# SELECT inner_product('[1,2,3]','[2,3,4]');
  ```
- 比较大小（<、>、<=、>=、<>、=）
  ```
  gaussdb=# SELECT floatvector('[1,2,3]') < floatvector('[1,2,3]');
  gaussdb=# SELECT boolvector('[T,F,T]') = boolvector('[T,T,T]');
  ```
  说明：boolvector仅支持等于操作，不支持大小比较。

2.4 创建和管理索引

2.4.1 向量索引 GsIVFFLAT

GsIVFFLAT索引是针对小数据量（万~百万级别）下的向量检索，支持在已创建含向量数据的列上创建单列索引，以及支持删除索引、支持数据变更后的索引自动更新。

前置条件

- 数据库的GUC参数enable_vectordb设置为on。
- 使用GsIVFFLAT索引时，数据不宜少于1*10^4条或超过2*10^6条，数据量过少时建议不建立索引，数据量过多时建议使用其他向量索引。

使用建议

- N条数据的表建议取ivf_nlist=4*sqrt(N)，如果数据量较大需设置ivf_nlist2，则建议ivf_nlist2=sqrt(ivf_nlist)。
- GUC参数gsivfflat_probes建议设置为索引超参ivf_nlist的10%，根据所需查询精度和查询时延适当增加或降低gsivfflat_probes，查询时间与gsivfflat_probes/ivf_nlist的比值线性相关。同样，如创建索引时设置了ivf_nlist2，则GUC参数gsivfflat_secondary_probes建议设置为索引超参ivf_nlist2的10%，根据所需查询精度和查询时延适当增加或降低gsivfflat_secondary_probes，查询时间与gsivfflat_secondary_probes/ivf_nlist2的比值线性相关。
- 建议将所有需要查询的数据提前插入表中，最后再建立索引；如果决定先建立索引再导入数据，在数据导入完成后需要重建索引。

- 建立索引后如果数据变化（含插入、删除、更新）超过20%会产生数据分布漂移，导致查询精度下降与查询时间不稳定，建议每更新10%~20%数据重建一次索引。
- 建立索引时数据条目数若低于nlist取值，会将nlist视为1创建索引，并弹出提示。此时查询与全表扫描无异，包括准确率与时延。为使索引达到预期性能，需在数据导入完成后重建索引。需要注意，对表进行truncate操作后，GsIVFFLAT索引也会弹出相同的提示，使索引处于上述状态。如希望使索引重新可用，需要在数据重新导入后重建索引。

参数说明

参数名称	取值说明	参数描述
ivf_nlist	**取值范围：** 1~65535 **默认值：** 100	倒排列表的数量。
ivf_nlist2	**取值范围：** 0~65535 **默认值：** 0	二级倒排列表数量。当ivf_nlist2取值为0或1时，不会创建二级倒排列表。
num_parallels	**取值范围：** 1~64 **默认值：** 1	构建索引并行计算数量，仅在NPU构建索引时生效。

须知

在向量索引中一次性进行大量的删除或者更新操作，或者频繁的删除或者更新操作，可能会导致索引质量下降，索引膨胀，搜索时间变长的问题。

查询索引状态

使用函数gs_ivfflat_inspect('ivfflat_index_name')可查询当前GsIVFFLAT索引状态，包括索引磁盘使用、聚簇的数据分布等信息。每条信息的返回类型为record，需要用SELECT * FROM语句获取，函数参数为索引名，由建索引时指定或者按照默认规则生成，该名可以使用\d table_name或者从系统表pg_index进行查询。

对于包含向量索引的分区表，该函数会返回所有分区下索引的状态，查询某一具体分区的索引请使用gs_ivfflat_inspect('ivfflat_index_name', 'index_partition_name')只显示该分区索引的索引状态，索引分区名可以通过查询系统表pg_partition获取。

须知

在使用查询索引状态函数分析索引数据过程时若有大量DML操作可能会造成分析结果有误差，执行过程中数据变化量若小于原数据总量的5%则输出结果可信度较高。

频繁调用gs_ivfflat_inspect函数可能会影响并发操作的性能。建议两次调用该函数的间隔时间不小于 10 秒，由此保证对并发操作性能的劣化影响控制在5%以内。

示例

```
--创建表
gaussdb=# CREATE TABLE "test" ("id" BIGINT PRIMARY KEY, "repr" floatvector (128));

--随机插入数据
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# INSERT INTO test SELECT i,floatvector(float_random_array(128,100)) FROM
generate_series(1,10000) as i;

--创建索引
gaussdb=# CREATE INDEX testivfflat ON test USING GSIVFFLAT(repr L2) WITH (IVF_NLIST = 256);

--查询索引状态
gaussdb=# SELECT * FROM gs_ivfflat_inspect('testivfflat');
    inspect_item      |                           inspect_info
-----------------------+--------------------------------------------------------------------
 FixedPageBlk          | MetaPage 0, ListGuard 1, BktGuard 2, BktScanGuard 3, ExtendGuard 4
 TotalPageNum          | 858
 UsedPageNum           | 858
 Utilization Rate      | 100.000000%
 FirstLevelListNum     | 256
 SecondLevelListNum    | 0
 FirstLvlTupCntMedian  | 39
 FirstLvlTupCntAverage | 39
 FirstLvlTupCntStdDev  | 24.3591
 FirstLvlTupCntMax-K   | 104,102,97,97,91
 FirstLvlTupCntMin-K   | 1,2,2,2,2
 CanScanWithNPU        | false
(11 rows)

--向量ANN检索
gaussdb=# set gsivfflat_probes = 25;
gaussdb=# SELECT "id" FROM "test"  ORDER BY "repr" <-> '[
23.0,0.0,0.0,0.0,1.0,5.0,43.0,114.0,3.0,0.0,0.0,0.0,14.0,121.0,120.0,78.0,81.0,4.0,0.0,0.0,31.0,126.0,23.0,18.0,126.0
,12.0,0.0,0.0,0.0,1.0,0.0,10.0,0.0,0.0,0.0,0.0,8.0,29.0,96.0,43.0,0.0,0.0,0.0,0.0,1.0,81.0,126.0,44.0,126.0,1.0,0.0,0.0,
1.0,45.0,66.0,96.0,126.0,0.0,0.0,0.0,1.0,16.0,12.0,63.0,1.0,2.0,0.0,0.0,11.0,40.0,26.0,0.0,5.0,20.0,28.0,1.0,0.0,17.0,
36.0,5.0,126.0,45.0,10.0,1.0,0.0,2.0,12.0,29.0,126.0,6.0,0.0,0.0,2.0,110.0,96.0,46.0,18.0,13.0,0.0,0.0,3.0,5.0,1.0,2.0
,29.0,50.0,30.0,7.0,8.0,3.0,0.0,1.0,55.0,24.0,14.0,5.0,9.0,15.0,8.0,10.0,10.0,1.0,0.0,0.0,19.0,79.0,16.0,4.0
]'  LIMIT 10;

--插入数据
gaussdb=# INSERT INTO test(id,repr) VALUES(99999999,'[
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0
]');
gaussdb=# SELECT COUNT(*) FROM test;

--更新数据
gaussdb=# UPDATE test SET repr = '[
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0
]' WHERE id = 99999999;
gaussdb=# SELECT * FROM test WHERE id = 99999999;

--删除数据
gaussdb=# DELETE FROM test WHERE id = 99999999;
gaussdb=# SELECT COUNT(*) FROM test;

--清理数据
gaussdb=# VACUUM test;

--删除函数
gaussdb=# DROP FUNCTION float_random_array;

--删除索引
```

```
gaussdb=# DROP INDEX testivfflat;

--删除表
gaussdb=# DROP TABLE test;
```

2.4.2 向量索引 GsDiskANN

GsDiskANN是针对大数据量（千万/亿级别）的向量检索，支持在已创建含向量数据的列上创建单列索引，并支持删除索引、支持数据变更后的索引自动更新。

前置条件

- 数据库正常运行，用户正常登录。
- 已创建带有向量类型的表，且表中已有少量规模行数的数据（1w行以内）。
- 数据库的GUC参数enable_vectordb设置为on。

使用约束

- GsDiskANN索引仅支持浮点类型（FLOAT）。
- GsDiskANN索引支持的向量长度上限为4096维。
  - 使用原始向量时，上限为1024维。
  - 启用pq/lvq量化压缩时，上限为4096维，此时需要设置enable_vector_copy=false以及subgraph_count > 0。
  - 启用pq量化压缩时，需设置pq_nseg和pq_nclus，以保证量化后的向量大小不超过页面大小8000 bytes。具体计算方法：将pq_nclus向上取到16/256，分别占用0.5/1 byte，再乘以pq_nseg，得到量化向量大小。

使用建议

- 在需要加速查询的向量属性列上创建GsDiskANN类型索引。
- 参数pq_nseg控制的是PQ创建段数，此段数必须要满足的条件是向量维度要能够被段数整除。段数设置的越大，占用空间越大，但是距离计算越准确；因此经验推荐设置规则如下：
  - 当向量维度小于512维度的时候，建议设置段数和维度相同。
  - 当向量维度高于512维度小于等于1024维度的时候，建议设置为维度的一半（能整除的情况下）。
- 参数pq_nclus控制的是PQ每段的分类数，此参数控制的是每个片段的映射聚类个数，此参数必须要满足少于数据行数，片段聚类设置越大，距离计算精度越高；向量维度1024维度以内，推荐值是16。
- 参数num_parallels控制的是构建索引时候的并发度，在内存充足（能够放下全部索引），Intel Xeon 5代以上处理器环境上，这个参数在30并发以内可以多核跑满，建议设置值是50；num_parallels参数受到数据库内bgworker最大值的限制，这个值是64，数据库进程同时申请占用的bgworker数量超过这个值会报错；特别需要注意的是，多会话同时创建索引的场景，应该避免创建过多的线程总数。
- 从已有数据表构建索引时，数据量充足的情况下，参数enable_pq推荐值是true，以便使用PQ计算邻居节点距离，避免大量磁盘（或者共享内存）访问，以加速构建。

说明

- 如果在表数据不足的情况下，强行创建enable_pq=true的索引，索引创建会上报告警，并且自动切换为enable_pq=false的方式进行创建；当使用未经过PQ压缩的gsdiskann索引时，会导致性能存在劣化，如果希望重新创建enable_pq=true的索引，请保证数据量充足的情况下进行REINDEX操作。在数据库进行表TRUNCATE操作、表级别备份恢复等过程中会发生数据不足导致无法完成创建索引的问题，完成之后建议重新构建索引，保证使用PQ压缩优化的索引效率。
- 自动扩展分区场景中如果使用enable_pq=true参数构建索引，可能会产生较多数量的告警，属于正常提醒；如果不希望提示告警，可以将索引enable_pq参数设置为false。

- 从已有数据表构建索引时，数据量不足的情况下，参数enable_pq的值必须设置为false，此场景不使用PQ计算距离。
- 从已有数据表构建索引时，参数using_clustering_for_parallel在开启之后可以缓解并发冲突，但是存在聚类不均衡导致长尾的风险，默认是关闭的。
- 新增数据格式必须与创建的数据表各项类型一致，向量数据维度也需要一致。
- 删除数据需要指定待删除数据的搜索条件，支持单点和批量删除。
- 由于存储长度限制，GsDiskANN图中的每个节点出度有限制，因此无法保证全图连通性，用户查询离群点的时候可能会发生查询返回结果少于指定TopK的情景，如果需要保证输出数量，请改用GsIVFFLAT索引。
- GsDiskANN索引大小预计为原数据占用磁盘的10-50倍，建索引前请提前确定磁盘容量。
- 由于GsDiskANN每次更新一条向量之后需要更新多个索引页数据，因此记录的日志量较大，可能会频繁出现LOG: checkpoints are occurring too frequently的日志打印，属于正常现象；如果不希望打印，在资源充足的情况下可以增加参数“checkpoint_segments”（参见《参考》中“数据库运行参数说明 > GUC参数说明 > 预写式日志 > 检查点”章节）的大小。
- 从大量已有数据表构建索引时，预计索引大小若大于shared_buffers时，会导致构建时间大幅度延长，该场景下可以通过设置subgraph_count参数将索引分片构建，加速构建时间，但会降低构建出索引的检索召回率。参数subgraph_count将索引分成数值设置的片数，每片大小需小于maintenance_work_mem，若用户规格充足，建议在当前连接中将maintenance_work_mem设置为不低于最大可用内存的33%后构建索引。如果maintenance_work_mem不足会在构建初报错并给出估计的最低建议规格。在规格不足时可以尝试增加subgraph_count分片数量，或者设置为0由磁盘存储格式构建。

说明

- subgraph_count分片数量越高，建成索引的检索召回率越低，设置数值不建议超过10。
- 在使用astore时，默认情况下params_adaptive为true。若subgraph_count为0，数据库会自动将其调整为 1。若params_adaptive设置为false，则subgraph_count默认为0时会触发报错。此时，需显式设置subgraph_count为大于0的值，或将params_adaptive设置为true以避免错误。
- 创建索引时的内存需求超过maintenance_work_mem参数指定的内存大小时，会产生警告，并将subgraph_count参数值视为0进行处理。请注意，此情况可能导致索引构建时长超出预期，如果用户无法接受此行为，请终止构建流程，并将maintenance_work_mem参数值设置为大于抛出预警中描述的内存需求值的范围并进行重新构建。内存需求的计算是统计当前数据量的大小，并根据统计数据量估算需要的内存大小，此提示值为统计值，大小可能会波动，建议根据实际情况在提示值基础上适当增大，将GUC参数maintenance_work_mem设置为合适大小。
- maintenance_work_mem属于user_set的GUC参数。如果用户直接设置maintenance_work_mem的大小对自动扩展分区表创建索引，产生的自治事务的maintenance_work_mem值不会发生改变，当创建索引时的内存需求超过之前设置的maintenance_work_mem参数指定的内存大小时，会产生警告，并将subgraph_count参数值视为0进行处理。建议用户使用reload修改参数。

- 参数enable_neighbor_embedded用于指定是否允许邻居向量密集存储，即每个点是否存储所有邻居的向量副本，该参数仅在enable_pq=true且subgraph_count>0时生效。启用此参数会导致索引大小发生显著膨胀，且查询所需的缓存也会相应增加。然而，当向量长度较短时，查询速度可能会显著提升。
- 参数enable_vector_copy用于指定是否存储原始向量的副本，即索引中是否存储原始向量的副本，该参数仅在enable_pq=true且subgraph_count>0时生效。 查询时，路由过程使用原始向量，可以少量提升查询精度；查询时，重排过程使用原始向量，可以少量提升查询速度。
- 参数build_with_quantized_vector用于指定构建时是否使用编码向量，即是否使用量化向量构建，该参数仅在enable_pq=true且subgraph_count>0时生效。启用此参数，会使用量化向量替代原始向量进行构建，减少构建所需的内存，可能会导致构建精度下降，且构建速度可能上升或下降。
- 参数quantization_type用于指定量化算法类型，选择量化方法，取值范围是{pq, lvq}，默认值是pq，该参数仅在enable_pq=true时生效。
- 参数lvq_nclus用于指定LVQ的量化向量的长度和质量，该参数仅在enable_pq=true且quantization_type=lvq时生效。
- 参数graph_degree用于指定图的连通度，控制图索引中每个节点的出度，该参数仅在subgraph_count>0时生效，推荐值为96。
- 参数build_instruction_set用于指定构建所用的指令集，默认值为AUTO。当subgraph_count=1时，支持{AUTO、NONE、SSE4、AVX2、AVX512F、AVX512BWDQ、NEON}。当subgraph_count=0时，仅支持{AUTO、SSE4、NEON}。取值范围与效果如下：
  - SSE4：可用于几乎所有的X86平台，但性能较差。
  - AVX2：可用于大部分X86平台，性能较好，支持相对快速的基于量化向量构建。
  - AVX512F：可用于部分X86平台，部分指令计算较快。
  - AVX512BWDQ：可用于高性能X86平台，性能较优，支持最高性能的基于量化向量构建。
  - NEON：可用于几乎所有ARM平台，但性能较差。
  - AUTO：基于用户机器的信息，自动寻找所支持的性能较优的指令集并显式使用。

- NONE：不显式使用指令集，由编译器决定使用何种指令集进行优化。

说明

在升级观察期时创建向量索引GsDiskANN，新版本参数取值仅支持默认值，既往版本参数取值无额外限制。

参数说明

GsDiskANN仅支持floatvector。

floatvector支持欧式距离（L2）、余弦距离（Cosine）的计算。对于维度为dim的向量，其中L2的索引标量范围是-9.223E+18/sqrt(dim)~9.223E+18/sqrt(dim)，Cosine的索引标量范围同样是-9.223E+18/sqrt(dim)~9.223E+18/sqrt(dim)。超出此范围时，如果运算过程中有数据溢出情况，会提示报错并返回。

在505.2.1版本之前，建立索引时标量超出此范围时，不会有显式报错，但仍然不推荐超出此范围，运算过程中超过3.402E+38的数据会被截断至3.402E+38；在505.2.1版本及之后，建立索引时标量超出此范围时，如果运算过程中有数据溢出情况，会提示报错并返回。

距离类型	标量约束
L2	(-9.223E+18/sqrt(dim), 9.223E+18/sqrt(dim))
Cosine	(-9.223E+18/sqrt(dim), 9.223E+18/sqrt(dim))

说明

对于维度相同的向量a、b，在余弦距离（Cosine）的计算过程中，如果a、b有任意一个向量为零向量，那么余弦距离的计算结果为NaN，一般情况需要避免零向量参与余弦距离的计算。

参数名称	取值说明	参数描述
pq_nseg	**取值范围：** 1~65535 **默认值：** 1	原始矢量量化分片个数。该参数在enable_pq=true，且quantization_type='pq'时有效。
pq_nclus	**取值范围：** 2~256 **默认值：** 16	每个量化分片的中心点个数。该参数在enable_pq=true，且quantization_type='pq'时有效。
queue_size	**取值范围：** 64~1000 **默认值：** 100	构建磁盘时候选列表的大小。
num_parallels	**取值范围：** 1~64 **默认值：** 10	构建索引并行计算数量。
using_clustering_for_parallel	**取值范围：** T/F **默认值：** F	是否使用集群进行并行构建（T为是，F为否）。该参数在subgraph_count=0时有效。

参数名称	取值说明	参数描述
lambda_for_balance	**取值范围：** 0~DBL_MAX **默认值：**0.00001	控制并行计算之间的平衡。
enable_pq	**取值范围：**T/F **默认值：**T	在构建和搜索时是否使用PQ进行加速（T为是，F为否）。
subgraph_count	**取值范围：**0~32 **默认值：**0	控制索引分片构建数量。
enable_neighbor_embedded	**取值范围：**T/F **默认值：**F	每个点是否存储所有邻居的向量副本。 ● T：表示存储所有邻居的向量副本。 ● F：表示不存储所有邻居的向量副本。
enable_vector_copy	**取值范围：**T/F **默认值：**T	是否存储原始向量的副本。 ● T：表示存储原始向量的副本。 ● F：表示不存储原始向量的副本。
build_with_quantized_vector	**取值范围：**T/F **默认值：**F	构建时是否使用编码向量。 ● T：表示构建时使用编码向量。 ● F：表示构建时不使用编码向量。
quantization_type	**取值范围：**pq/lvq **默认值：**pq	量化算法类型。该参数在enable_pq=true时有效。
lvq_nclus	**取值范围：**2~128 **默认值：**128	LVQ的量化向量的长度和质量。该参数在enable_pq=true，且quantization_type='lvq'时有效。
graph_degree	**取值范围：**48~256 **默认值：**96	图索引中每个节点的出度。
build_instruction_set	**取值范围：**SSE4/ AVX2/ AVX512F/ AVX512BWDQ/ NEON/ AUTO/ NONE **默认值：**AUTO	指定构建所用的指令集。

参数名称	取值说明	参数描述
params_adaptive	取值范围：T/F 默认值：T	当maintenance_work_mem内存不足时，是否支持索引参数自适应降级（例如：子图构建降级为磁盘构建）。 ● T：表示支持。 ● F：表示不支持。

须知

在向量索引中一次性进行大量的删除或者更新操作，或者频繁的删除或者更新操作，可能会导致索引质量下降，索引膨胀，搜索时间变长的问题。

查询索引健康状态

使用函数gs_diskann_inspect('diskann_index_name')可查询索引健康状态，包括索引磁盘使用、PQ压缩准确度、索引图连接健康度、Vacuum状态等信息。每条信息的返回类型为record，需要用select * from语句获取，函数参数为索引名，由建索引时指定或者按照默认规则生成，该名可以使用\d table_name或者从系统表pg_index进行查询。对于包含向量索引的分区表，该函数会返回所有分区下索引的状态，查询某一具体分区的索引请使用gs_diskann_inspect('diskann_index_name', 'index_partition_name')只显示该分区索引的健康状态，索引分区名可以通过查询系统表pg_partition获取。

须知

在使用查询索引健康状态函数分析索引数据过程时若有大量DML操作可能会造成分析结果有误差甚至分析函数执行失败并报错，执行过程中数据变化量若小于原数据总量的5%则输出结果可信度较高。在设置subgraph_count>1，build_with_quantized_vector=true，graph_degree<96等索引构建质量参数时，可能会导致索引构建质量下降，应谨慎调整。

示例

```
--创建表
gaussdb=# CREATE TABLE "test" ("id" BIGINT PRIMARY KEY, "repr" floatvector (128));

--随机插入数据
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# INSERT INTO test SELECT i,floatvector(float_random_array(128,100)) FROM
generate_series(1,1000) as i;

--创建索引
gaussdb=# CREATE INDEX diskann_sift1m_l2 on test USING GsDiskANN (repr L2) with (pq_nseg=128,
pq_nclus=16, num_parallels=50, enable_pq=true, using_clustering_for_parallel=false);

--向量ANN检索
gaussdb=# SELECT "id" FROM "test"  ORDER BY "repr" <-> '[
```

```
23.0,0.0,0.0,0.0,1.0,5.0,43.0,114.0,3.0,0.0,0.0,0.0,14.0,121.0,120.0,78.0,81.0,4.0,0.0,0.0,31.0,126.0,23.0,18.0,126.0
,12.0,0.0,0.0,0.0,1.0,0.0,10.0,0.0,0.0,0.0,0.0,8.0,29.0,96.0,43.0,0.0,0.0,0.0,0.0,1.0,81.0,126.0,44.0,126.0,1.0,0.0,0.0,
1.0,45.0,66.0,96.0,126.0,0.0,0.0,0.0,1.0,16.0,12.0,63.0,1.0,2.0,0.0,0.0,11.0,40.0,26.0,0.0,5.0,20.0,28.0,1.0,0.0,17.0,
36.0,5.0,126.0,45.0,10.0,1.0,0.0,2.0,12.0,29.0,126.0,6.0,0.0,0.0,2.0,110.0,96.0,46.0,18.0,13.0,0.0,0.0,3.0,5.0,1.0,2.0
,29.0,50.0,30.0,7.0,8.0,3.0,0.0,1.0,55.0,24.0,14.0,5.0,9.0,15.0,8.0,10.0,10.0,1.0,0.0,0.0,19.0,79.0,16.0,4.0
]'  LIMIT 10;

--插入数据
gaussdb=# INSERT INTO test(id,repr) VALUES(99999999,'[
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0
]');
gaussdb=# SELECT COUNT(*) FROM test;

--更新数据
gaussdb=# UPDATE test SET repr = '[
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0
]' WHERE id = 99999999;
gaussdb=# SELECT * FROM test WHERE id = 99999999;

--删除数据
gaussdb=# DELETE FROM test WHERE id = 99999999;
gaussdb=# SELECT COUNT(*) FROM test;

--清理数据
gaussdb=# VACUUM test;

--查询索引健康度
gaussdb=# SELECT * FROM gs_diskann_inspect('diskann_sift1m_l2');

--删除函数
gaussdb=# DROP FUNCTION float_random_array;

--删除索引
gaussdb=# DROP INDEX diskann_sift1m_l2 ;

--删除表
gaussdb=# DROP TABLE test;
```

2.4.3 向量标量混合索引 GsDiskANN

向量标量混合索引基于GsDiskANN基础上，增加了标量非数组过滤、标量数组过滤功能；同时具备向量索引GsDiskANN在大规模数据上检索、新增、删除等能力。

前置条件

- 数据库正常运行，用户正常登录。
- 已创建带有向量类型和标量类型的表。
- 数据库的GUC参数enable_vectordb设置为on。

使用约束

- 混合索引仅支持一个向量字段和最多31个标量字段，且第一个字段必须为向量类型字段。
- 混合索引不支持备机读。
- 混合索引支持的非数组标量数据类型：
  - 整数类型（TINYINT、SMALLINT、INTEGER、BIGINT）；
  - 任意精度类型（NUMERIC、DECIMAL）；

  - 浮点类型（FLOAT、DOUBLE）；
  - 布尔类型（BOOLEAN）；
  - 字符类型（CHAR、VARCHAR），VARCHAR必须带长度参数，CHAR和VARCHAR参数<=256；
  - 日期/时间类型（DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE）；
  - 枚举类型（ENUM）。
- 混合索引支持的数组标量数据类型：
  - 如果标量字段为数组类型，则数组维度为单维，且数组元素个数建议不超过128个；数组元素类型为上述支持的非数组标量数据类型；
  - 创建索引时，标量数组字段不能超过2个。
- 查询WHERE条件中包含索引标量，进行索引过滤需要遵循以下规则：
  - 非数组标量列支持比较运算符 <、<=、>、>=、=、IN；
  - 非数组标量列和数组标量列均支持逻辑运算符AND、OR；
  - 非数组标量列支持字符运算符LIKE且只支持前匹配；
  - 非数组标量列和数组标量列均支持IS NULL、IS NOT NULL；
  - 数组标量列支持操作符：=、&&、@>、<@。
- 查询WHERE条件中包含的非索引标量、不满足以上规则的索引标量，无法在向量标量混合索引内进行过滤，均在索引后过滤且召回率暂无保障。
- 混合索引不支持唯一索引、表达式索引、GLOBAL索引；支持多字段索引、LOCAL索引、部分索引。
- 针对分区表上的LOCAL索引，当查询语句涉及多个分区，命中多个分区索引时，可能会出现执行变慢的情况。
- 混合索引支持ustore存储和astore存储。
- 创建索引时，必须指定subgraph_count参数，且取值在[1,32]。其余索引参数设置请参见**向量索引GsDiskANN**。

须知

- 在升级观察期时禁止创建向量标量混合索引GsDiskANN。
- 非Oracle兼容的数据库下使用TINYINT存在隐式类型转换，影响向量标量混合索引的检索策略正确生成，建议采用如下方式规避该问题：
  1. 在非Oracle兼容的数据库下使用其他整型代替TINYINT。
  2. 使用Oracle兼容的数据库，即CREATE DATABASE mydb DBCOMPATIBILITY 'ORA'。

示例（向量标量混合检索非数组）

```
--创建表
gaussdb=# CREATE TABLE "test" ("id" BIGINT PRIMARY KEY,"repr" floatvector(128),"score" float);

--随机插入数据
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# CREATE OR REPLACE FUNCTION float_random(range int) RETURNS float AS $$BEGIN RETURN
(random() * range)::int;END;$$ LANGUAGE plpgsql;
gaussdb=# INSERT INTO test SELECT i,floatvector(float_random_array(128,100)),float_random(100) FROM
```

```
generate_series(1,10000) as i;

--创建索引
gaussdb=# SET maintenance_work_mem="2GB";
gaussdb=# CREATE INDEX diskann_sift1m_l2 on test USING GsDiskANN (repr L2, score) WITH
(pq_nseg=128, pq_nclus=16, num_parallels=32, enable_pq=true, subgraph_count = 1);

--向量标量混合检索
gaussdb=# SELECT "id" FROM "test" WHERE (score < 60) ORDER BY "repr" <->
'[23.0,0.0,0.0,0.0,1.0,5.0,43.0,114.0,3.0,0.0,0.0,0.0,14.0,121.0,120.0,78.0,81.0,4.0,0.0,0.0,31.0,126.0,23.0,18.0,126.
0,12.0,0.0,0.0,0.0,1.0,0.0,10.0,0.0,0.0,0.0,0.0,8.0,29.0,96.0,43.0,0.0,0.0,0.0,0.0,1.0,81.0,126.0,44.0,126.0,1.0,0.0,0.0
,1.0,45.0,66.0,96.0,126.0,0.0,0.0,0.0,1.0,16.0,12.0,63.0,1.0,2.0,0.0,0.0,11.0,40.0,26.0,0.0,5.0,20.0,28.0,1.0,0.0,17.0,
36.0,5.0,126.0,45.0,10.0,1.0,0.0,2.0,12.0,29.0,126.0,6.0,0.0,0.0,2.0,110.0,96.0,46.0,18.0,13.0,0.0,0.0,3.0,5.0,1.0,2.0
,29.0,50.0,30.0,7.0,8.0,3.0,0.0,1.0,55.0,24.0,14.0,5.0,9.0,15.0,8.0,10.0,10.0,1.0,0.0,0.0,19.0,79.0,16.0,4.0]' LIMIT
10;

--插入数据
gaussdb=# INSERT INTO test(id,repr,score) VALUES(99999999,'[
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0
]', 86.5);
gaussdb=# SELECT COUNT(*) FROM test;

--更新数据
gaussdb=# UPDATE test SET repr = '[
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0
]', score = 93 WHERE id = 99999999;
gaussdb=# SELECT * FROM test WHERE id = 99999999;

--删除数据
gaussdb=# DELETE FROM test WHERE id = 99999999;
gaussdb=# SELECT COUNT(*) FROM test;

--清理数据
gaussdb=# VACUUM test;

--查询索引健康度
gaussdb=# SELECT * FROM gs_diskann_inspect('diskann_sift1m_l2');

--删除函数
gaussdb=# DROP FUNCTION float_random_array;
gaussdb=# DROP FUNCTION float_random;

--删除索引
gaussdb=# DROP INDEX diskann_sift1m_l2;

--删除表
gaussdb=# DROP TABLE test;
```

示例（向量标量混合检索支持数组）

```
--创建表
gaussdb=# CREATE TABLE "test" ("id" BIGINT PRIMARY KEY,"repr" floatvector(128),"score" float,"e" INT[]);

--随机插入数据
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# CREATE OR REPLACE FUNCTION float_random(range int) RETURNS float AS $$BEGIN RETURN
(random() * range)::int;END;$$ LANGUAGE plpgsql;
gaussdb=# INSERT INTO test (id, repr, score, e) SELECT i,floatvector(float_random_array(128,
100)),float_random(100),float_random_array(128, 100)::INT[]
FROM generate_series(1, 10000) AS i;

--创建索引
gaussdb=# SET maintenance_work_mem="2GB";
```

```
gaussdb=# CREATE INDEX diskann_sift1m_l2_array on test USING GsDiskANN (repr L2, score, e) WITH
(pq_nseg=128, pq_nclus=16, num_parallels=32, enable_pq=true, subgraph_count = 1);

--向量标量混合检索
gaussdb=# SELECT "id","score" FROM "test" WHERE (score = 60) AND e @> ARRAY[1] ORDER BY "repr" <-
>
'[23.0,0.0,0.0,0.0,1.0,5.0,43.0,114.0,3.0,0.0,0.0,0.0,14.0,121.0,120.0,78.0,81.0,4.0,0.0,0.0,31.0,126.0,23.0,18.0,126.
0,12.0,0.0,0.0,0.0,1.0,0.0,10.0,0.0,0.0,0.0,0.0,8.0,29.0,96.0,43.0,0.0,0.0,0.0,0.0,1.0,81.0,126.0,44.0,126.0,1.0,0.0,0.0
,1.0,45.0,66.0,96.0,126.0,0.0,0.0,0.0,1.0,16.0,12.0,63.0,1.0,2.0,0.0,0.0,11.0,40.0,26.0,0.0,5.0,20.0,28.0,1.0,0.0,17.0,
36.0,5.0,126.0,45.0,10.0,1.0,0.0,2.0,12.0,29.0,126.0,6.0,0.0,0.0,2.0,110.0,96.0,46.0,18.0,13.0,0.0,0.0,3.0,5.0,1.0,2.0
,29.0,50.0,30.0,7.0,8.0,3.0,0.0,1.0,55.0,24.0,14.0,5.0,9.0,15.0,8.0,10.0,10.0,1.0,0.0,0.0,19.0,79.0,16.0,4.0]' LIMIT
10;

--插入数据
gaussdb=# INSERT INTO test(id,repr,score,e) VALUES(99999999,'[
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0
]', 86.5,ARRAY[1,2,3,99,98,100]);
gaussdb=# SELECT COUNT(*) FROM test;


--更新数据
gaussdb=# UPDATE test SET e = ARRAY[1,2,3],score = 93 WHERE id = 99999999;
gaussdb=# SELECT * FROM test WHERE id = 99999999;

--删除数据
gaussdb=# DELETE FROM test WHERE id = 99999999;
gaussdb=# SELECT COUNT(*) FROM test;

--清理数据
gaussdb=# VACUUM test;

--查询索引健康度
gaussdb=# SELECT * FROM gs_diskann_inspect('diskann_sift1m_l2_array');

--删除函数
gaussdb=# DROP FUNCTION float_random_array;
gaussdb=# DROP FUNCTION float_random;

--删除索引
gaussdb=# DROP INDEX diskann_sift1m_l2_array;

--删除表
gaussdb=# DROP TABLE test;
```

2.5 相似性搜索

通常使用距离来描述向量间的相似度。当前向量数据库提供欧式距离、余弦距离以及汉明距离三种距离计算方式。其中影响欧式距离的包括向量间的角度差和长度差两方面；余弦距离则只关注向量间的角度差别；汉明距离仅用来计算boolvector，计算向量中不同元素的数量。

欧式距离相似性搜索示例

计算向量间欧氏距离时，仅支持floatvector类型向量使用：操作符<->和系统函数l2_distance。

- l2_distance操作符（<->）
```
gaussdb=# CREATE TABLE t1(id int unique, repr floatvector(4));
gaussdb=# INSERT INTO t1 VALUES(0, '[30,12,12,25]');
gaussdb=# SELECT id, repr<-> '[1,1,3,2]' AS s FROM t1;
gaussdb=# SELECT floatvector('[1,2,3]')<->floatvector('[5,-1,3.5]');
```

- 系统函数（l2_distance）

```
gaussdb=# SELECT id, l2_distance(repr, '[1,1,3,2]') AS s FROM t1;
gaussdb=# SELECT l2_distance(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
```

  不定义的字符类型（unknown）可进行隐式转换

```
gaussdb=# SELECT l2_distance('[1,2,3]',  '[5,-1,3.5]');
```

- 相同维度的数据字段进行相似性查找，输出结果：

  向量索引只支持升序排序，当前特性向量索引不支持降序排列。

```
gaussdb=# SELECT id FROM t1 ORDER BY repr<-> '[1,1,3,2]' LIMIT 2;
```

  支持多表关联场景

```
gaussdb=# CREATE TABLE t2(id int unique, repr floatvector(4));
gaussdb=# INSERT INTO t2 VALUES(0, '[30,12,12,25]');
gaussdb=# SELECT t1.repr FROM t1 inner JOIN t2 ON t1.repr <-> t2.repr < 0.8;
```

  支持子查询场景

```
gaussdb=# SELECT t1.id FROM t1 WHERE t1.repr <-> (SELECT t2.repr FROM t2 LIMIT 1) < 0.8;
```

余弦距离相似性搜索示例

计算向量间余弦距离时，仅支持 floatvector 类型向量使用：操作符<+>和系统函数cosine_distance。

- 操作符（<+>）

```
gaussdb=# select id, repr<+> '[1,1,3,2]' as s from t1;
gaussdb=# select floatvector('[1,2,3]')<+>floatvector('[5,-1,3.5]');
```

- 系统函数（cosine_distance）

```
gaussdb=# select id, cosine_distance(repr, '[1,1,3,2]') as s from t1;
gaussdb=# select cosine_distance(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
```

  不定义的字符类型（unknown）可进行隐式转换

```
gaussdb=# select cosine_distance('[1,2,3]', '[5,-1,3.5]');
```

- 同维度的数据字段进行相似性查找，输出结果：

  向量索引只支持升序排序，当前特性向量索引不支持降序排列。

```
gaussdb=# select id from t1 order by repr<+> '[1,1,3,2]' limit 2;
```

  支持多表关联场景

```
gaussdb=# select t1.repr from t1 inner join t2 on t1.repr <+> t2.repr < 0.8;
```

  支持子查询场景

```
gaussdb=# select t1.id from t1 where t1.repr <+> (select t2.repr from t2 limit 1) < 0.8;
```

汉明距离相似性搜索示例

计算向量间汉明距离时，仅支持boolvector类型向量使用：操作符<#>和系统函数hamming_bool_distance。

- 操作符（<#>）

```
gaussdb=# DROP TABLE t2;
gaussdb=# CREATE TABLE t2(id int unique, repr boolvector(3));
gaussdb=# INSERT INTO t2 VALUES(1, '[1, 0, 1]');
gaussdb=# select id, repr<#> '[1,1,0]' as s from t2;
gaussdb=# select boolvector('[T,F,F]')<#>boolvector('[T,F,T]');
```

- 系统函数（hamming_bool_distance）

```
gaussdb=# select id, hamming_bool_distance(repr, '[1,1,0]') as s from t2;
gaussdb=# select hamming_bool_distance(boolvector('[T,F,F]'), boolvector('[T,F,T]'));
```

  不定义的字符类型（unknown）可进行隐式转换

```
gaussdb=# select hamming_bool_distance('[T,F,F]', '[T,F,T]');
```


- 同维度的数据字段进行相似性查找，输出结果：

  向量索引只支持升序排序，当前特性向量索引不支持降序排列。

  ```
  gaussdb=# select id from t2 order by repr<#> '[1,1,0]' limit 2;
  ```

  支持多表关联场景

  ```
  gaussdb=# CREATE TABLE t4(id int unique, repr boolvector(3));
  gaussdb=# INSERT INTO t4 VALUES(1, '[1, 0, 1]');
  gaussdb=# select t2.repr from t2 inner join t4 on t2.repr <#> t4.repr < 3;
  ```

  支持子查询场景

  ```
  gaussdb=# select t2.id from t2 where t2.repr <#> (select t4.repr from t4 limit 1) < 3;
  ```

2.6 相似文档检索

对于非结构化数据，例如文档类的数据进行检索，传统的数据检索引擎会根据文档中文字片段与查询的相似程度，按照相关性排名，然后召回数据。向量数据库支持相似文档的查询检索，文档相似评分根据BM25算法族实现。

2.6.1 使用方式

向量数据库支持对数据库表中文本列属性建立索引并进行基于BM25算法族的相似性排名的检索召回，索引列属性支持范围包括文本TEXT类型和文本数组TEXT[]类型；索引中文本数组类型数据中数组的每一个元素都会作为一个单词进行BM25算法的词倒排建模，而文本数据会由数据库内置的bm25_tokenize函数进行自动文本分词成单词的文本数组后建立词倒排模型，目前只保证中英文文本的分词有效性。

本小节介绍向量数据库中文本索引的基本功能，包括创建表、创建文本索引、使用索引进行相似文档召回、查询索引信息等操作。

如要使用外部分词器，请使用TEXT[]属性列创建文本索引，并导入经过外部分词得到的文本数组数据。

步骤1 创建包含需要做相似文本检索的TEXT[]属性列的数据表。

```
gaussdb=# CREATE TABLE st_information (st_id SERIAL PRIMARY KEY, st_name TEXT[], st_email TEXT[]);
```

步骤2 在数据表中插入数据。

```
gaussdb=# INSERT INTO st_information(st_name, st_email) VALUES
('{"Joe", "Root"}', '{"joe@xyz.com", "ceo@xyz.com", "common-domain@xyz.com"}'),
('{"Tim", "David"}', '{"tim@xyz.com", "tim2@xyz.com", "joe@xyz.com", "common-domain@xyz.com"}'),
('{"Random", "Dude"}', '{"ceo@xyz.com", "tim2@xyz.com", "joe@xyz.com", "common-domain@xyz.com",
"repeated@xyz.com"}'),
('{"Why", "You"}', '{"ceo@xyz.com", "tim2@xyz.com", "joe@xyz.com", "common-domain@xyz.com",
"repeated@xyz.com", "repeated@xyz.com", "repeated@xyz.com"}'),
('{"Some", "Really", "Long", "Name"}', '{"some-really-long-email-of-the-user@xyz.com", "common-
domain@xyz.com"}'),
('{"Boring", "Boris"}', '{"common-domain@xyz.com"}');
```

步骤3 构建文本全文检索索引，设置构建并发度为16。

```
gaussdb=# CREATE INDEX st_information_st_email_bm25_index ON st_information USING BM25(st_email)
WITH (num_parallels=16);
```

步骤4 使用索引按关键字"common-domain@xyz.com"检索bm25分数最高的文档。

```
-- 设置检索算法
gaussdb=# SET bm25_ranking_metric = 0;
-- 进行检索
gaussdb=# SELECT /*+ indexscan(st_information st_information_st_email_bm25_index) */ * FROM
st_information ORDER BY st_email ### '{common-domain@xyz.com}' DESC LIMIT 1;
```

通过bm25_ranking_metric设置BM25算法族文档相似评分算法，支持的评分方式如下：

- 0：BM25_OKAPI (默认值)
- 1：BM25_ATIRE
- 2：BM25_L
- 3：BM25_PLUS
- 4：TF_IDF

步骤5 通过视图查询当前索引的页面占用信息、统计信息。

```
gaussdb=# SELECT * FROM gs_bm25_inspect('st_information_st_email_bm25_index');
```

----结束

如要使用内置分词器，请使用TEXT属性列创建文本索引，文本数据会在索引内部自动调用gs_bm25_tokenize函数进行分词。

步骤1 创建包含TEXT属性列的数据表。

```
gaussdb=# CREATE TABLE bm25_test (id INT, text TEXT);
```

步骤2 在数据表中插入数据，text列插入文本数据。

```
gaussdb=# INSERT INTO bm25_test VALUES
(1, '向量数据库'),
(2, 'bm25索引'),
(3, 'gsivfflat索引'),
(4, 'gsdiskann索引'),
(5, '向量标量混合索引');
```

步骤3 构建BM25索引。

```
gaussdb=# CREATE INDEX bm25_test_text_idx ON bm25_test USING BM25(text);
```

步骤4 使用BM25索引检索最相关的两篇文档。

```
gaussdb=# SELECT /*+ indexscan(bm25_test bm25_test_text_idx) */ *, text ### '向量数据库' AS score FROM bm25_test ORDER BY score DESC LIMIT 2;
```

----结束

2.6.2 约束

- BM25文本索引不支持创建分区表全局分区索引（GPI），也不支持local partition index分区索引创建。
- BM25文本索引不支持创建Unique索引。
- BM25文本索引不支持Clusterable索引。
- BM25文本索引不支持创建多列索引。
- BM25文本索引仅能构建在TEXT列或者TEXT[]列，或类型为TEXT或者TEXT[]的表达式上。
- BM25文本索引不支持dblink。
- BM25文本索引构建期间需要部分工作内存，大小由maintenance_work_mem设置，如果遇到内存不足将首先尽可能通过任务分块完成执行，如无法执行任务则索引无法完成构建；推荐规格为shared_buffers=32GB, maintenance_work_mem=8GB。
- BM25文本索引的分词仅支持UTF-8编码的中文、英文，不支持其他语种语言。
- BM25文本索引仅支持建在TEXT[]字段或TEXT字段（将使用内部分词）上，不支持用户在数据库内自定义UDF并在分词函数上构建表达式索引。
- Ustore只支持RCR索引格式，不支持PCR索引格式。

- 不支持并行构建，不支持在线创建索引。
- BM25文本索引不支持使用表达式的查询。
- BM25文本索引查询只支持DESC降序查询，不支持ASC升序查询。
- BM25文本索引查询需要通过HINT指定使用bm25索引扫描。
- BM25文本索引不支持闪回查询。
- BM25文本索引不支持备机读，不支持按需读。
- 当查询内容全部为BM25索引对应词典所定义的停用词，查询结果为空。
- 在分区表中的BM25文本索引只能返回查询在该分区中的相似分数，分数计算完全独立于其他分区的数据统计信息，所以跨分区的BM25查询相似分数不具有比较意义。
- BM25文本索引支持数据插入实时生效，为获得良好插入性能，建议同时插入的线程数控制在30以下。
- 混合标量的文本相关性检索，为提升检索性能，建议禁用gplan、aplan。
- 混合标量的文本相关性检索，为提升极低选择率（0.001%）下的性能，建议对标量列建立btree索引，同时查询语句不再使用hint，该用法不支持gplan、aplan，不支持分区表。

2.6.3 自定义 Tokenweight 分词词典

Tokenweight词典主要用于中文分词，支持根据词权重，进行基于隐马尔可夫模型分布算法的段落分词。GaussDB中预定义默认中文分词词典，可通过系统表PG_TS_DICT（详细请参见《参考》中“系统表和系统视图 > 系统表 > 其他系统表 > PG_TS_DICT”章节）查看。Tokenweight词典为相似文档召回检索的BM25索引提供分词功能，加强中文相关文档检索的准确性。用户可根据业务需求自定义词典关键词权重，并在建立BM25索引时通过tokenize_dict参数指定词典名使索引按照自定义词典进行文档关键词切分。

Tokenweight词典支持定义关键词、同义词和停用词。其中关键词用于提升文本分词质量进而提高相似文档检索的准确性。同义词用于扩展查询语句涉及文档范围，提高检索的召回率。停用词用于过滤查询语句中无意义的词，加速检索时延。用户可以通过**BM25索引**，建索引语法指定该索引使用的词典，不指定则会设置为系统默认词典，其中配置了常见中文词汇和标准中英文停用词集。

注意事项

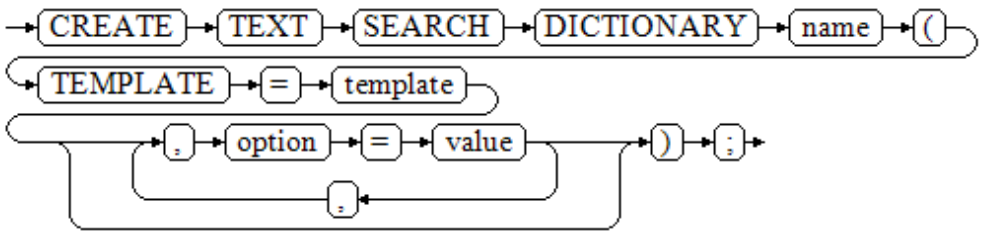
- 具有SYSADMIN权限的用户可以执行创建词典操作，创建该词典的用户自动成为其所有者。
- 临时模式（pg_temp）下不允许创建词典。
- 创建或修改词典之后，任何对于用户自定义的词典定义文件的修改，将不会影响到数据库中的词典。如果需要在数据库中使用这些修改，需使用ALTER语句更新对应词典的定义文件。
- 用户自定义词典不支持通过gs_dump/gs_dumpall工具进行导入导出，需要用户重新手动创建词典。
- 全文检索词典相关语法操作受GUC参数tsearch_options限制。

语法格式

```
CREATE TEXT SEARCH DICTIONARY name (
    TEMPLATE = template
```



```
    [, { option = value } [, ... ]]
);
```



参数说明

- **name**

  要创建的词典的名称（可指定模式名，否则在当前模式下创建）。

  取值范围：符合**标识符命名规范**的字符串，且最大长度不超过63个字符。

- **template**

  模板名。

  取值范围：系统表PG_TS_TEMPLATE（详细信息请参见《参考》中“系统表和系统视图 > 系统表 > 其他系统表 > PG_TS_TEMPLATE”章节）中定义的模板，本特性只涉及模板Tokenweight。

- **option**

  参数名。模板Tokenweight对应参数及说明如下。

  Tokenweight词典

  - USEDEFAULT

    参数数值类型为布尔值，创建词典时是否使用BM25系统默认词典内容作为词典数据。设置为true时词典创建后将会自动导入系统默认设置常见的中文单词和停用词，为false时则内容为空。如果手动指定文件则该参数会被忽略，但不建议通过文件创建词典，推荐词典创建流程为创建词典，随后增量添加自定义的词典内容。

  - TOKENFILE

    词权重文件名，默认后缀名为tokw。文件格式为词权重和同义词相似度列表，每行可以表示词权重或者同义词相似度，词典处理时，文件中空行或者开头为“#”的注释以及每行开头和结尾的空格会被忽略。

    - 词权重格式为token word [: token weight]。token weight可以为小数或者整数，默认值为3，中间“:”作为分隔符，词权重越高分词时越容易将该字符串分成一个词。
    - 同义词支持两种格式：
      1) 多同义词：token word 1 :: token word 2 [:: token word 3 [...]]。表示所有token words均互为同义词，相似度配置为0.8，过多的同义词组设置可能会造成包含同义词成员的BM25检索时延的增长。
      2) 单向同义词：token word 1 -> token word 2 [: similarity]。表示token word 2是token word 1的同义词，similarity可以为0-1之间的数值表示其相似度，默认值为0.8。注意该定义中同义词相似度只表示单向同义，A为B的同义词不代表B为A的同义词。

  - STOPFILE

    停用词表文件名，默认后缀名为stop，文件格式要求与Simple类型词典的停用词文件相同。

- FILEPATH

  词典定义文件所在目录。可以指定为本地目录，格式为file:// absolute_path。默认值为预定义词典文件所在目录。FILEPATH参数必须和STOPFILE或者TOKENFILE参数同时指定，不允许单独指定。

  说明

  词典定义文件的文件名仅支持小写字母、数据、下划线混合。

增量添加词典内容

使用函数gs_ts_dict_add_definition(regdictionary, "char", text[])可以增量添加词典内容。

- 第一个参数为词典OID，可以直接输入词典名字符串由数据库内部做自动转换。
- 第二个参数表示该次增量词典内容添加的内容类型，'t'表示关键词或者同义词，'s'表示停用词。
- 第三个参数是添加的词典内容，类型为文本数组，数组中每条数据应单独表示一条词典内容定义，定义格式请参见**TOKENFILE**。

  说明

  该函数效果等同于DDL语句，运行后会清空所有数据库连接的词典缓存，重新加载词典时会造成BM25文本索引增删查操作大幅度变慢，不建议在线使用。

标识符命名规范

标识符的命名需要遵守如下规范：

- 标识符需要为字母（a-z）、下划线（_）、数字（0-9）或美元符号（$）。
- 标识符必须以字母（a-z）或下划线（_）开头。

  说明

  - 此命名规范为建议项，非强制项。
  - 特殊情况下可以使用双引号规避特殊字符报错。

3 相关参考

3.1 向量数据类型

向量数据类型包含floatvector和boolvector两种类型。

- floatvector数据类型是指多维数据中含有的数据为float类型，例如[1.0,3.0,11.0,110.0,62.0,22.0,4.0]。

  说明

  - floatvector成员仅支持单精度。单精度范围为-3.402E+38~+3.402E+38，6位十进制数字精度。
  - floatvector不支持NULL、Nan、Inf作为元素，当向量中含有NULL值，数据库会报错。
  - floatvector不能为NULL，当插入、更新或转换NULL值作为向量数据时，数据库会报错。

- boolvector数据类型是指多维数据中含有的数据为bool类型，例如[0,0,0,0,0,1,0,0,1,0,0,0]。

  说明

  - boolvector成员可使用t(T)\f(F)、y(Y)\n(N)、1\0、true\false、yes\no、on\off等方式表达布尔型数据。
  - boolvector不支持NULL、Nan、Inf作为元素，当向量中含有NULL值，数据库会报错。
  - boolvector不能为NULL，当插入、更新或转换NULL值作为向量数据时，数据库会报错。

向量类型的使用

向量类型的使用示例如下：

```
-- 创建包含向量类型的表，同时设定数据维度。建表时向量类型必须要指定维度。
gaussdb=# CREATE TABLE t1(id int unique, repr floatvector(4));
gaussdb=# CREATE TABLE t2(id int unique, repr boolvector(3));

-- 插入数据。
gaussdb=# INSERT INTO t1 VALUES(0, '[30, 12, 12, 25]');
gaussdb=# INSERT INTO t2 VALUES(1, '[1, 0, 1]');

-- 删除表。
gaussdb=# DROP TABLE t1;
gaussdb=# DROP TABLE t2;
```


3.2 向量数据库函数与操作符

3.2.1 向量距离计算接口

l2_distance

功能说明：计算两个向量的欧式距离。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：float8

代码示例：

```
gaussdb=# SELECT l2_distance(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT l2_distance('[1,2,3]',  '[5,-1,3.5]');
```

vector_l2_squared_distance

功能说明：获得两个向量的欧式距离的平方。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：float8

代码示例：

```
gaussdb=# SELECT vector_l2_squared_distance(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_l2_squared_distance('[1,2,3]',  '[5,-1,3.5]');
```

cosine_distance

功能说明：计算两个向量的余弦距离。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：float8

代码示例：

```
gaussdb=# SELECT cosine_distance(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT cosine_distance('[1,2,3]', '[5,-1,3.5]');
```

vector_spherical_distance

功能说明：计算两个归一化向量的球面距离（余弦夹角的弧度制表示）。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：float8

代码示例：

```
gaussdb=# SELECT vector_spherical_distance('[1,0,0,0]', '[0,0,0,1]');
```

说明

如果输入向量非归一化，无法获得正确的计算结果。

3.2.2 向量操作函数接口

向量操作函数实现的功能包括：向量大小比较、向量加法、向量减法、向量按位乘法等。

inner_product

功能说明：计算两个向量的内积。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：float8

代码示例：

```
gaussdb=# SELECT inner_product(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT inner_product('[1,2,3]', '[5,-1,3.5]');
```

vector_negative_inner_product

功能说明：计算两个向量的负内积。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：float8

代码示例：

```
gaussdb=# SELECT vector_negative_inner_product(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_negative_inner_product('[1,2,3]', '[5,-1,3.5]');
```

vector_dims

功能说明：返回floatvector向量的维度值。

入参类型：floatvector

出参类型：int4

代码示例：

```
gaussdb=# SELECT vector_dims(floatvector('[1,2,3]'));
gaussdb=# SELECT vector_dims('[1,2,3]');
```

vector_norm

功能说明：返回向量的L2范数。

入参类型：floatvector

出参类型：float8

代码示例：

```
gaussdb=# SELECT vector_norm(floatvector('[1,2,3]'));
gaussdb=# SELECT vector_norm('[1,2,3]');
```

vector_add

功能说明：计算两个向量相加。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：floatvector

代码示例：

```
gaussdb=# SELECT vector_add(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_add('[1,2,3]', '[5,-1,3.5]');
```

vector_sub

功能说明：计算两个向量相减。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：floatvector

代码示例：

```
gaussdb=# SELECT vector_sub(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_sub('[1,2,3]', '[5,-1,3.5]');
```

vector_lt

功能说明：比较向量大小，向量1是否小于向量2。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT vector_lt(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_lt('[1,2,3]', '[5,-1,3.5]');
```

vector_le

功能说明：比较向量大小，向量1是否小于等于向量2。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：BOOLEAN

代码示例：


```
gaussdb=# SELECT vector_le(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_le('[1,2,3]', '[5,-1,3.5]');
```

vector_eq

功能说明：比较向量是否相等。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT vector_eq(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_eq('[1,2,3]', '[5,-1,3.5]');
```

vector_ne

功能说明：比较两个向量是否不等。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT vector_ne(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_ne('[1,2,3]', '[5,-1,3.5]');
```

vector_ge

功能说明：比较向量大小，向量1是否大于等于向量2。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT vector_ge(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_ge('[1,2,3]', '[5,-1,3.5]');
```

vector_gt

功能说明：比较向量大小，向量1是否大于向量2。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT vector_gt(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_gt('[1,2,3]', '[5,-1,3.5]');
```


vector_cmp

功能说明：比较向量大小。

入参1的类型：floatvector

入参2的类型：floatvector

出参类型：int4

代码示例：

```
gaussdb=# SELECT vector_cmp(floatvector('[1,2,3]'), floatvector('[5,-1,3.5]'));
gaussdb=# SELECT vector_cmp('[1,2,3]', '[5,-1,3.5]');
```

vector_accum

功能说明：返回向量累加。

入参1的类型：anyarray

入参2的类型：floatvector

出参类型：anyarray

代码示例：

```
--系统函数，不推荐使用，若需使用，数组元素类型必须为float8类型。
gaussdb=# SELECT vector_accum(array[cast(3 as float8),1,2,3], floatvector('[5,-1,3.5]'));
```

vector_combine

功能说明：合并向量。

入参1的类型：anyarray

入参2的类型：anyarray

出参类型：anyarray

代码示例：

```
--系统函数，不推荐使用，若需使用，数组元素类型必须为float8类型。
gaussdb=# SELECT vector_combine(array[cast(1 as float8),2,3], array[cast(1 as float8),2,3]);
```

vector_avg

功能说明：平均向量。

入参类型：anyarray

出参类型：floatvector

代码示例：

```
--系统函数，不推荐使用，若需使用，数组元素类型必须为float8类型。
gaussdb=# SELECT vector_avg(array[cast(1 as float8),2,3]);
```

bool_vector_dims

功能说明：返回boolvector向量的维度值。

入参类型：boolvector

出参类型：int4

代码示例：

```
gaussdb=# SELECT bool_vector_dims(boolvector('[1,1,1]'));
```

bool_vector_cmp

功能说明：比较布尔向量大小。

入参1的类型：boolvector

入参2的类型：boolvector

出参类型：int4

代码示例：

```
gaussdb=# SELECT bool_vector_cmp('[1,0,0]','[1,1,0]');
gaussdb=# SELECT bool_vector_cmp('[1,1,0]','[1,1,0]');
```

bool_vector_eq

功能说明：比较bool向量是否一致。

入参1的类型：boolvector

入参2的类型：boolvector

出参类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT bool_vector_eq(boolvector('[1,1,1]'), boolvector('[1,1,1]'));
gaussdb=# SELECT bool_vector_eq('[1,1,1]', '[1,1,1]');
```

3.2.3 向量函数和操作符

floatvector支持向量类型和数组类型之间的数据转换，同时支持特定格式的字符串转换成向量类型。

array<->floatvector：向量数据类型可以和相对应的数组类型进行自由转换；floatvector向量的成员数据类型为浮点型。

string ->floatvector：注意字符串格式，使用中括号（[]）或者花括号（{}）包含数组，元素间使用逗号（,）隔开。

floatvector

功能说明：

场景1：数组数据转换为向量数据。

入参类型：anyarray

出参类型：floatvector

代码示例：

```
gaussdb=# SELECT floatvector(ARRAY[1,2,9.3]);
```

场景2：floatvector转换，对维度进行检测。

入参1的类型：floatvector

入参2的类型：integer

出参类型：floatvector

代码示例：

```
gaussdb=# SELECT floatvector(floatvector('[1,2,9.3]'),3);
```

vector_to_array

功能说明：向量数据转换为数组数据。

入参类型：floatvector

出参类型：real[]

代码示例：

```
gaussdb=# SELECT vector_to_array(floatvector('[1,2,3]'));
```

text_to_vector

功能说明：字符型数据转换为向量数据。

入参类型：cstring

出参类型：floatvector

代码示例：

```
gaussdb=# SELECT text_to_vector('[1,2,9.3]');
```

vector_in

功能说明：floatvector输入转换函数（文本格式）。

入参1的类型：cstring

入参2的类型：oid

入参3的类型：int4

出参类型：floatvector

代码示例：

```
gaussdb=# SELECT vector_in('[1,2,3]',701,3);
```

vector_out

功能说明：floatvector输出转换函数（文本格式）。

入参类型：floatvector

出参类型：cstring

代码示例：

```
gaussdb=# SELECT vector_out(floatvector('[1,2,3]'));
```


vector_send

功能说明：floatvector输入转换函数（二进制格式）。

入参类型：floatvector

出参类型：bytea

代码示例：

```
gaussdb=# SELECT vector_send(floatvector('[1,2,3]'));
```

vector_recv

功能说明：floatvector输出转换函数（二进制格式）。

入参1的类型：integer

入参2的类型：oid

入参3的类型：int4

出参类型：floatvector

代码示例：

```
--系统函数，无法sql调用。
```

vector_typmod_in

功能说明：floatvector输入类型修改符函数。

入参类型：cstring[]

出参类型：integer

代码示例：

```
gaussdb=# SELECT vector_typmod_in( '{1}' );
```

说明

- 向量数据类型成员仅支持单精度。
- 向量间计算仅支持相同维度，如果维度不同将报错。
- floatvector支持向量加减操作，点乘操作由函数（inner_product）完成。
- 建表时向量类型必须要指定维度；当插入数据的维度与设置的数据维度不一致时，系统会报错。
- 需要注意如果数据表中已经插入数据，变换向量维度时需要表中数据符合维度设定，否则数据库报错；当向量属性已创建向量索引，属性的数据类型无法切换到其他数据类型，数据库会报错。

boolvector

场景1：

功能说明：数组数据转换为向量数据。

入参类型：anyarray

出参类型：boolvector

代码示例：

```
gaussdb=# SELECT boolvector(ARRAY[1,0,1]);
```

场景2：

功能说明：boolvector转换，对维度进行检测。

入参1的类型：boolvector

入参2的类型：integer

出参类型：boolvector

代码示例：

```
gaussdb=# SELECT boolvector(boolvector('[1,0,1]'),3);
```

boolvector_to_array

功能说明：向量数据转换为数组数据。

入参类型：boolvector

出参类型：anyarray

代码示例：

```
gaussdb=# SELECT boolvector_to_array(boolvector('[1,0,1]'));
```

text_to_boolvector

功能说明：字符型数据转换为向量数据。

入参类型：cstring

出参类型：boolvector

代码示例：

```
gaussdb=# SELECT text_to_boolvector('[1,1,1]');
```

bool_vector_in

功能说明：boolvector输入转换函数（文本格式）。

入参1的类型：cstring

入参2的类型：oid

入参3的类型：int4

出参类型：boolvector

代码示例：

```
gaussdb=# SELECT bool_vector_in('[1,1,1]',701,3);
```

bool_vector_out

功能说明：boolvector输出转换函数（文本格式）。

入参类型：boolvector

出参类型：cstring

代码示例：

```
gaussdb=# SELECT bool_vector_out(boolvector('[1,1,1]'));
```

bool_vector_send

功能说明：boolvector输入转换函数（二进制格式）。

入参类型：boolvector

出参类型：bytea

代码示例：

```
gaussdb=# SELECT bool_vector_send(boolvector('[1,1,1]'));
```

bool_vector_recv

功能说明：boolvector输出转换函数（二进制格式）。

入参1的类型：internal

入参2的类型：oid

入参3的类型：int4

出参类型：boolvector

代码示例：

```
--系统函数，无法sql调用。
```

bool_vector_typmod_in

功能说明：boolvector输入类型修改符函数。

入参类型：cstring[]

出参类型：integer

代码示例：

```
gaussdb=# SELECT bool_vector_typmod_in( '{1}' );
```

说明

- 向量数据类型不支持Null、Nan、Inf作为元素，当向量中含有NULL值，数据库会报错。
- 插入向量数据类型时，不支持NULL作为插入值，当插入NULL值作为向量数据时，数据库会报错。
- boolvector类型的元素可使用t(T)\f(F)、y(Y)\n(N)、1\0、true\false、yes\no、on\off等方式表达布尔型数据。
- boolvector仅支持等于操作，不支持大小比较。

gs_vector_index_options

功能说明：显示相关向量索引的超参取值。

入参类型：text

出参类型：text

代码示例：

```
--创建表。
gaussdb=# CREATE TABLE t1 (id int unique,repr floatvector(960));
--插入数据：
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# INSERT INTO t1 SELECT i,floatvector(float_random_array(960,100)) FROM
generate_series(1,1000) as i;
--创建索引:
gaussdb=# CREATE INDEX test1v on t1 using gsdiskann (repr l2) with
(pq_nseg=120,pq_nclus=64,queue_size=120,num_parallels=30,enable_pq=true,using_clustering_for_parallel=f
alse);

gaussdb=# SELECT gs_vector_index_options('test1v');
```

gs_diskann_inspect

功能说明：查询索引健康状态，包括索引磁盘使用、PQ压缩准确度、索引图连接健康度、Vacuum状态等信息。

场景1：

入参类型：text

出参类型：record

代码示例：

```
-- 创建表
gaussdb=# CREATE TABLE t1 (id int unique,repr floatvector(960));
-- 插入数据
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# INSERT INTO t1 SELECT i,floatvector(float_random_array(960,100)) FROM
generate_series(1,1000) as i;
-- 创建向量索引
gaussdb=# CREATE INDEX test1v on t1 using gsdiskann (repr l2) with
(pq_nseg=120,pq_nclus=64,queue_size=120,num_parallels=30,enable_pq=true,using_clustering_for_parallel=f
alse);
-- 显示索引状态
gaussdb=# SELECT * FROM gs_diskann_inspect('test1v');
```

场景2：

入参1的类型：text

入参2的类型：text

出参类型：record

代码示例：

```
-- 创建分区表
gaussdb=# CREATE TABLE t1 (id int unique,repr floatvector(960)) partition by hash(id) (partition p1,
partition p2, partition p3);
-- 插入数据
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# INSERT INTO t1 SELECT i,floatvector(float_random_array(960,100)) FROM
generate_series(1,1000) as i;
-- 创建分区向量索引
```



```
gaussdb=# CREATE INDEX test1v on t1 using gsdiskann (repr l2) local with
(pq_nseg=120,pq_nclus=64,queue_size=120,num_parallels=30,enable_pq=true,using_clustering_for_parallel=f
alse);
-- 查询分区名和分区索引名
SELECT oid,relname,parttype,parentid,boundaries FROM pg_partition WHERE parentid = 't1'::regclass::oid;
SELECT oid,relname,parttype,parentid,boundaries,indextblid FROM pg_partition WHERE parentid =
'test1v'::regclass::oid;
-- 显示分区索引状态
gaussdb=# SELECT * FROM gs_diskann_inspect('test1v', 'p1_repr_idx');
```

gs_ivfflat_inspect

功能说明：查询GsIvfflat索引状态，包括固定页面号、索引磁盘使用、聚簇中向量数据分布等信息。

场景1：

入参类型：text

出参类型：record

代码示例：

```
-- 创建表
gaussdb=# CREATE TABLE t1 (id int unique,repr floatvector(960));
-- 插入数据
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# INSERT INTO t1 SELECT i,floatvector(float_random_array(960,100)) FROM
generate_series(1,1000) as i;
-- 创建向量索引
gaussdb=# SET maintenance_work_mem='128MB';
gaussdb=# CREATE INDEX test1v on t1 using gsivfflat (repr l2) with (ivf_nlist=10);
-- 显示索引状态
gaussdb=# SELECT * FROM gs_ivfflat_inspect('test1v');
```

场景2：

入参1的类型：text

入参2的类型：text

出参类型：record

代码示例：

```
-- 创建分区表
gaussdb=# CREATE TABLE t1 (id int unique,repr floatvector(960)) partition by hash(id) (partition p1,
partition p2, partition p3);
-- 插入数据
gaussdb=# CREATE OR REPLACE FUNCTION float_random_array(dims int,range int) RETURNS float[] AS $
$BEGIN RETURN ARRAY(SELECT (random() * range)::int FROM generate_series(1, dims));END;$$
LANGUAGE plpgsql;
gaussdb=# INSERT INTO t1 SELECT i,floatvector(float_random_array(960,100)) FROM
generate_series(1,1000) as i;
-- 创建分区向量索引
gaussdb=# SET maintenance_work_mem='128MB';
gaussdb=# CREATE INDEX test1v on t1 using gsivfflat (repr l2) LOCAL with (ivf_nlist=10);
-- 查询分区名和分区索引名
gaussdb=# SELECT oid,relname,parttype,parentid,boundaries FROM pg_partition WHERE parentid =
't1'::regclass::oid;
gaussdb=# SELECT oid,relname,parttype,parentid,boundaries,indextblid FROM pg_partition WHERE parentid
= 'test1v'::regclass::oid;
-- 显示分区索引状态
gaussdb=# SELECT * FROM gs_ivfflat_inspect('test1v', 'p1_repr_idx');
```

针对向量计算的操作符，主要集中在衡量向量间相似度，以及向量的二元计算符：

<->

功能说明：计算两向量（floatvector）之间欧氏距离（L2）。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：double precision

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') <->  floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'<-> floatvector('[1,1,3,2]');
```

<+>

功能说明：计算两向量（floatvector）之间余弦距离（COSINE）。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：double precision

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') <+>  floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'<+> floatvector('[1,1,3,2]');
```

<#>

功能说明：计算两向量（boolvector）之间的汉明距离（Hamming distance）。

左参数类型：boolvector

右参数类型：boolvector

返回值类型：double precision

代码示例：

```
gaussdb=# SELECT boolvector('[1,0,1,0]') <#>  boolvector('[1,1,1,0]');
gaussdb=# SELECT '[1,0,1,0]'<#> boolvector('[1,1,1,0]');
```

+

功能说明：计算两个维度相同的向量按位相加。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：floatvector

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') + floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'+ floatvector('[1,1,3,2]');
```

-

功能说明：计算两个维度相同的向量按位相减。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：floatvector

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') + floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'+ floatvector('[1,1,3,2]');
```

<

功能说明：比较两个维度相同的向量字典序的大小关系。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') < floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'< floatvector('[1,1,3,2]');
```

<=

功能说明：比较两个维度相同的向量字典序的大小关系。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') <= floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'<= floatvector('[1,1,3,2]');
```

>

功能说明：比较两个维度相同的向量字典序的大小关系。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') > floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'> floatvector('[1,1,3,2]');
```

>=

功能说明：比较两个维度相同的向量字典序的大小关系。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') >= floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'>= floatvector('[1,1,3,2]');
```

=

场景1：

功能说明：判断两个维度相同的向量是否相等。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') = floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'= floatvector('[1,1,3,2]');
```

场景2：

功能说明：判断两个维度相同的布尔向量是否一致。

左参数类型：boolvector

右参数类型：boolvector

返回值类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT boolvector('[1,0,1,0]') =  boolvector('[1,1,1,0]');
gaussdb=# SELECT '[1,0,1,0]' =  boolvector('[1,1,1,0]');
```

<>

功能说明：判断两个维度相同的向量是否不相等。

左参数类型：floatvector

右参数类型：floatvector

返回值类型：BOOLEAN

代码示例：

```
gaussdb=# SELECT floatvector('[1,1,3,2]') <> floatvector('[1,1,3,2]');
gaussdb=# SELECT '[1,2,3,2]'<> floatvector('[1,1,3,2]');
```

说明

若向量元素值过大，在计算距离时，会出现中间计算结果溢出单精度范围的情况，导致距离结果为NAN或INF。

3.2.4 相似文档排序召回检索函数和操作符

###

场景1：

功能说明：基于BM25算法族计算两个文本间的相似度，只对使用BM25索引的查询有效。

- 左参数表示创建BM25索引的索引列名。
- 右参数表示查询文本。

左参数类型：text

右参数类型：text

返回值类型：double precision

代码示例：

```
-- 建表及BM25索引
gaussdb=# CREATE TABLE t1(_id TEXT UNIQUE, title TEXT, texts TEXT, metadata TEXT) WITH
(storage_type=astore);
gaussdb=# CREATE INDEX "bm25_idx1" ON "t1" USING bm25 ("texts");
-- 执行检索
gaussdb=# SELECT /*+ indexscan(t1 bm25_idx1) */ _id, texts ### 'drop table t1;' AS SCORE FROM t1
ORDER BY SCORE desc LIMIT 10;
```

场景2：

功能说明：基于BM25算法族计算两个分词文本数组间的相似度，只对使用BM25索引的查询有效。

- 左参数表示创建BM25索引的索引列名。
- 右参数表示查询文本数组。

左参数类型：text[]

右参数类型：text[]

返回值类型：double precision

代码示例：

```
-- 建表及BM25索引
gaussdb=# CREATE TABLE st_information (st_id SERIAL PRIMARY KEY, st_name TEXT[], st_email TEXT[]);
gaussdb=# CREATE INDEX st_information_st_email_bm25_index ON st_information USING bm25(st_email);
-- 执行检索
gaussdb=# SELECT /*+ indexscan(st_information, st_information_st_email_bm25_index) */ st_id, st_email
### '{common-domain@xyz.com}' AS score FROM st_information ORDER BY SCORE desc LIMIT 10;
```

gs_ts_dict_add_definition

功能说明：增量添加词典内容。

- 第一个参数为词典OID，可以直接输入词典名字符串由数据库内部做自动转换。
- 第二个参数表示该次增量词典内容添加的内容类型，'t'表示关键词或者同义词，'s'表示停用词。
- 第三个参数是添加的词典内容，类型为文本数组，数组中每条数据应单独表示一条词典内容定义。

说明

该函数效果等同于DDL语句，运行后会清空所有数据库连接的词典缓存，重新加载词典时会造成BM25文本索引增删查操作大幅度变慢，不建议在线使用。

入参类型：regdictionary, "char", text[]

出参类型：BOOLEAN

代码示例：

```
gaussdb=# CREATE TEXT SEARCH DICTIONARY test_dict (
    template = tokenweight,
    usedefault = false
);
gaussdb=# call gs_ts_dict_add_definition('test_dict', 't', array['高斯数据库:50']);
-- 对于定义了多关键词的情况，建议使用的导入方式如下：
gaussdb=# do $$
DECLARE
    tempwords TEXT;
begin
    create temp table tokenwords (t text);
    insert into tokenwords values ('单词一'), ('单词二'), ('单词三'), ('单词四'), ('单词五');
    call gs_ts_dict_add_definition('test_dict', 't', (select array_agg(t) into tempwords from tokenwords));
end$$;
gaussdb=# SELECT gs_bm25_tokenize('高斯数据库', 'test_dict');
 gs_bm25_tokenize
------------------
  {高斯数据库}
(1 row)
```

gs_bm25_tokenize

功能说明：使用指定词典对输入文本进行分词操作，返回被分割的单词文本数组。

- 第一个入参表示需要分词的文本。
- 第二个入参指定分词使用的词典。

入参类型：text, cstring(默认值"pg_catalog.simple_cn_segmentation")

出参类型：text[]

代码示例：

```
gaussdb=# SELECT gs_bm25_tokenize('高斯数据库');
gaussdb=# SELECT gs_bm25_tokenize('高斯数据库', 'pg_catalog.simple_cn_segmentation');
```

gs_bm25_inspect

功能说明：显示索引磁盘使用和内部词倒排信息等数据。

- 入参表示索引名。

入参类型：text

出参类型：record

代码示例：

```
-- 建表及BM25索引
gaussdb=# CREATE TABLE st_information (st_id SERIAL PRIMARY KEY, st_name TEXT[], st_email TEXT[]);
gaussdb=# CREATE INDEX st_information_st_email_bm25_index ON st_information USING bm25(st_email);
-- 执行检索
gaussdb=#  SELECT * FROM gs_bm25_inspect('st_information_st_email_bm25_index');
```


gs_bm25_distance_text

功能说明：返回bm25文档相似分数，只在使用BM25索引检索时有效。

- 入参一表示创建BM25索引的索引列名。
- 入参二表示查询文本。

入参类型：text, text

出参类型：double precision

代码示例：

```
-- 建表及BM25索引
gaussdb=# CREATE TABLE t1(_id TEXT UNIQUE, title TEXT, texts TEXT, metadata TEXT) WITH
(storage_type=astore);
gaussdb=# CREATE INDEX "bm25_idx1" ON "t1" USING bm25 ("texts");
-- 执行检索
gaussdb=# SELECT /*+ indexscan(t1, bm25_idx1) */ _id, gs_bm25_distance_text(texts, 'drop table t1;') AS
SCORE FROM t1 ORDER BY texts ### 'drop table t1;' desc LIMIT 10;
```

gs_bm25_distance_textarr

功能说明：返回bm25文档相似分数，只在使用BM25索引检索时有效。

- 入参一表示创建BM25索引的索引列名。
- 入参二表示查询文本数组。

入参类型：text[], text[]

出参类型：double precision

代码示例：

```
-- 建表及BM25索引
gaussdb=# CREATE TABLE st_information (st_id SERIAL PRIMARY KEY, st_name TEXT[], st_email TEXT[]);
gaussdb=# CREATE INDEX st_information_st_email_bm25_index ON st_information USING bm25(st_email);
-- 执行检索
gaussdb=# SELECT /*+ indexscan(st_information, st_information_st_email_bm25_index) */ *,
gs_bm25_distance_textarr(st_email, '{common-domain@xyz.com}') AS score FROM st_information ORDER
BY st_email ### '{common-domain@xyz.com}' desc LIMIT 10;
```

gs_bm25_docid_info

功能说明：返回指定BM25索引中，ctid对应的docid信息。

- 入参一表示BM25索引oid（对于LOCAL分区索引，表示每个分区中索引oid）。
- 入参二表示ctid中的block number。
- 入参三表示ctid中的offset number。
- 出参包含两列数据，第一列列名为dfx_item，表示返回项名称，第二列列名为dfx_info，表示返回项内容，具体说明如表3-1所示：

表 3-1 返回值说明

dfx_item	dfx_info
index_name	索引名。
doc	查询的doc文档。

dfx_item	dfx_info
ctid	查询的ctid。
docid	当前doc文档对应的docid。

说明

不建议在业务期间使用该接口，可能会导致BM25文本索引增删操作效率降低。

入参类型：oid, uint4, uint2

出参类型：record

代码示例：

非分区索引。

```
-- 建表及BM25索引
gaussdb=# SET current_schema=public;
gaussdb=# DROP TABLE IF EXISTS t01;
gaussdb=# CREATE TABLE t01(c1 int, c2 text);
gaussdb=# INSERT INTO t01 VALUES
(1, '向量数据库'),
(2, 'bm25索引'),
(3, 'gsivfflat索引'),
(4, 'gsdiskann索引'),
(5, '向量标量混合索引');
gaussdb=# CREATE INDEX idx ON t01 USING bm25(c2) WITH(num_parallels=16);
-- 查询索引oid，方法1
gaussdb=# SELECT oid FROM pg_class WHERE relname='idx' AND relnamespace=(SELECT oid FROM
pg_namespace WHERE nspname='public');
  oid
-------
 17578
(1 row)

-- 查询索引oid，方法2
gaussdb=# SELECT 'idx'::regclass::oid;
  oid
-------
 17578
(1 row)

-- 查询元组ctid
gaussdb=# SELECT ctid FROM t01 WHERE c1=2;
 ctid
-------
 (0,2)
(1 row)

-- 查询ctid对应的docid信息
gaussdb=# SELECT * FROM gs_bm25_docid_info(17578, 0, 2);
  dfx_item  | dfx_info
------------+----------
 index_name | idx
 doc        | bm25索引
 ctid       | (0,2)
 docid      | 1
(4 rows)
```

LOCAL分区索引，仅查询索引oid方式与上述示例有差异。

```
-- 建分区表及BM25 LOCAL分区索引
gaussdb=# SET current_schema=public;
gaussdb=# DROP TABLE IF EXISTS t01;
```



```
gaussdb=# CREATE TABLE IF NOT EXISTS t01(c1 INT, c2 text)
PARTITION BY LIST (c1) AUTOMATIC
(
    PARTITION p1 VALUES(1,2,3,4,5,6,7,8,9,10),
    PARTITION p2 VALUES(11,12,13,14,15,16,17,18,19,20),
    PARTITION p3 VALUES(21,22,23,24,25,26,27,28,29,30)
);
gaussdb=# INSERT INTO t01 VALUES(generate_series(1,30), '向量数据库');
gaussdb=# CREATE INDEX idx ON t01 USING bm25(c2) LOCAL WITH(num_parallels=16);
-- 查询每个分区中索引oid，表t01有三个分区，返回三个oid
gaussdb=# SELECT oid,relname FROM pg_partition WHERE parttype='x' AND parentid='idx'::regclass::oid;
  oid  |  relname
-------+-----------
 17608 | p3_c2_idx
 17607 | p2_c2_idx
 17606 | p1_c2_idx
(3 rows)

-- 查询p1分区元组ctid
gaussdb=# SELECT ctid FROM t01 WHERE c1=2;
 ctid
-------
 (0,2)
(1 row)

-- 查询ctid对应的docid信息
gaussdb=# SELECT * FROM gs_bm25_docid_info(17606, 0, 2);
  dfx_item  |  dfx_info
------------+------------
 index_name | p1_c2_idx
 doc        | 向量数据库
 ctid       | (0,2)
 docid      | 1
(4 rows)
```

gs_bm25_document_info

功能说明：返回指定BM25索引中，docid对应的正排信息。

- 入参一表示BM25索引oid（对于LOCAL分区索引，表示每个分区中索引oid）。
- 入参二表示docid。
- 出参包含两列数据，第一列列名为dfx_item，表示返回项名称，第二列列名为dfx_info，表示返回项内容，具体说明如表3-2所示：

表 3-2 返回值说明

dfx_item	dfx_info
index_name	索引名。
ctid	docid对应的ctid。
doc_index_info_is_alive	当前docid对应索引信息是否存在。
token_count_in_index	当前文档在索引中包含的token数量。

说明

不建议在业务期间使用该接口，可能会导致BM25文本索引增删操作效率降低。

入参类型：oid, uint4

出参类型：record

代码示例：

非分区索引（分区索引oid查询方式请参见**gs_bm25_docid_info**函数示例）。

```
-- 建表及BM25索引
gaussdb=# SET current_schema=public;
gaussdb=# DROP TABLE IF EXISTS t01;
gaussdb=# CREATE TABLE t01(c1 int, c2 text);
gaussdb=# INSERT INTO t01 VALUES
(1, '向量数据库'),
(2, 'bm25索引'),
(3, 'gsivfflat索引'),
(4, 'gsdiskann索引'),
(5, '向量标量混合索引');
gaussdb=# CREATE INDEX idx ON t01 USING bm25(c2) WITH(num_parallels=16);
-- 查询docid为0的文档正排信息
gaussdb=# SELECT * FROM gs_bm25_document_info('idx'::regclass::oid, 0);
        dfx_item          | dfx_info
--------------------------+----------
 index_name               | idx
 docid                    | 0
 ctid                     | (0,1)
 doc_index_info_is_alive | true
 token_count_in_index     | 2
(5 rows)
```

gs_bm25_tokenhash_info

功能说明：返回指定BM25索引的哈希容器信息。

- 入参表示BM25索引oid（对于LOCAL分区索引，表示每个分区中索引oid）。
- 出参包含两列数据，第一列列名为dfx_item，表示返回项名称，第二列列名为dfx_info，表示返回项内容，具体说明如**表3-3**所示：

表 3-3 返回值说明

dfx_item	dfx_info
index_name	索引名。
hash_bucket_size	哈希容器bucket大小。
total_token_count	索引中总单词数。
bucket_count_in_use	使用中的bucket数量。
empty_bucket_count	空的bucket数量。
hash_bucket_load_factor	哈希容器负载因子。
conflict_bucket_count	冲突的bucket数量。
conflict_bucket_ratio	冲突的bucket比例。
bucket_length	bucket中链表的平均、中位数、最大长度。

说明

不建议在业务期间使用该接口，可能会导致BM25文本索引增删操作效率降低。

入参类型：oid

出参类型：record

代码示例：

非分区索引（分区索引oid查询方式请参见**gs_bm25_docid_info**函数示例）。

```
-- 建表及BM25索引
gaussdb=# SET current_schema=public;
gaussdb=# DROP TABLE IF EXISTS t01;
gaussdb=# CREATE TABLE t01(c1 int, c2 text);
gaussdb=# INSERT INTO t01 VALUES
(1, '向量数据库'),
(2, 'bm25索引'),
(3, 'gsivfflat索引'),
(4, 'gsdiskann索引'),
(5, '向量标量混合索引');
gaussdb=# CREATE INDEX idx ON t01 USING bm25(c2) WITH(num_parallels=16);
-- 查询指定BM25索引的哈希容器信息
gaussdb=# SELECT * FROM gs_bm25_tokenhash_info('idx'::regclass::oid);
        dfx_item         |                                dfx_info
-------------------------+-----------------------------------------------------------------------
 index_name              | idx
 hash_bucket_size        | 10000000
 total_token_count       | 8
 bucket_count_in_use     | 8
 empty_bucket_count      | 9999992
 hash_bucket_load_factor | 0.000080%
 conflict_bucket_count   | 0
 conflict_bucket_ratio   | 0.000000%
 bucket_length           | (avg, med, max) length of the list in a bucket: (0.000001, 1, 1)
(9 rows)
```

gs_bm25_token_info

功能说明：返回指定BM25索引的token信息。

- 入参一表示BM25索引oid（对于LOCAL分区索引，表示每个分区中索引oid）。
- 入参二表示要查询的文档字符串。
- 出参包含两列数据，第一列列名为token_name，表示文档在索引中包含的token字符串，第二列列名为token_id，表示token的id，返回行数不定。

说明

不建议在业务期间使用该接口，可能会导致BM25文本索引增删操作效率降低。

入参类型：oid, text

出参类型：record

代码示例：

非分区索引（分区索引oid查询方式请参见**gs_bm25_docid_info**函数示例）。

```
-- 建表及BM25索引
gaussdb=# SET current_schema=public;
gaussdb=# DROP TABLE IF EXISTS t01;
gaussdb=# CREATE TABLE t01(c1 int, c2 text);
gaussdb=# INSERT INTO t01 VALUES
(1, '向量数据库'),
(2, 'bm25索引'),
(3, 'gsivfflat索引'),
(4, 'gsdiskann索引'),
(5, '向量标量混合索引');
```



```
gaussdb=# CREATE INDEX idx ON t01 USING bm25(c2) WITH(num_parallels=16);
-- 查询文档"向量数据库"在索引中的token信息
gaussdb=# SELECT * FROM gs_bm25_token_info('idx'::regclass::oid, '向量数据库');
 token_name | token_id
------------+----------
 向量       | 0
 数据库     | 1
(2 rows)
```

gs_bm25_posting_info

功能说明：返回指定BM25索引中，tokenid对应的倒排信息。

- 入参一表示BM25索引oid（对于LOCAL分区索引，表示每个分区中索引oid）。
- 入参二表示tokenid。
- 出参包含两列数据，第一列列名为dfx_item，表示返回项名称，第二列列名为dfx_info，表示返回项内容，具体说明如表3-4所示：

表 3-4 返回值说明

dfx_item	dfx_info
index_name	索引名。
tokenid	查询的tokenid。
inverted_info_total_size	倒排信息中包含的总文档数。
uncompressed_inverted_info_size	倒排信息中未压缩的文档数。
uncompressed_inverted_info	倒排表<docid, appearance（词频）, doc_len（文档中的单词数）>。
compressed_inverted_info_size	倒排信息中压缩的文档数。
compressed_inverted_info	倒排表<docid, appearance, doc_len>。
...	剩余压缩块中的倒排信息。

说明

不建议在业务期间使用该接口，可能会导致BM25文本索引增删操作效率降低。

入参类型：oid, text

出参类型：record

代码示例：

非分区索引（分区索引oid查询方式请参见gs_bm25_docid_info函数示例）

```
-- 建表及BM25索引
gaussdb=# SET current_schema=public;
gaussdb=# DROP TABLE IF EXISTS t01;
gaussdb=# CREATE TABLE t01(c1 int, c2 text);
gaussdb=# INSERT INTO t01 VALUES
(1, '向量数据库'),
(2, 'bm25索引'),
(3, 'gsivfflat索引'),
```



```
(4, 'gsdiskann索引'),
(5, '向量标量混合索引');
gaussdb=# CREATE INDEX idx ON t01 USING bm25(c2) WITH(num_parallels=16);
-- 查询tokenid为0的单词在索引中的倒排信息
gaussdb=# SELECT * FROM gs_bm25_posting_info('idx'::regclass::oid, 0);
            dfx_item             |       dfx_info
---------------------------------+----------------------
 index_name                      | idx
 tokenid                         | 0
 inverted_info_total_size        | 2
 uncompressed_inverted_info_size | 2
 uncompressed_inverted_info      | <0, 1, 2> <4, 1, 4>
 compressed_inverted_info_size   | 0
 compressed_inverted_info        | NA
(7 rows)
```

3.3 向量索引

实现了部分向量索引访问方法，比如创建、维护、检索等。创建索引格式请参见《参考》中“SQL参考 > SQL语法 > C > CREATE INDEX”章节。向量索引不支持UNIQUE。

- 重建索引请参见《参考》中“SQL参考 > SQL语法 > R > REINDEX”章节。
- 删除索引请参见《参考》中“SQL参考 > SQL语法 > D > DROP INDEX”章节。
- **GLOBAL**
  分区表GLOBAL索引不支持。
- **DESC**
  不支持。
- **UNIQUE**
  不支持。
- **COLLATE collation**
  不支持。
- **DEDUPLICATION**
  不支持复合索引。
- **USING method**
  指定创建索引的方法。
  新增以下索引类型：
  gsivfflat：gsivfflat索引，针对向量数据的倒排索引。
  gsdiskann：gsdiskann索引，针对向量数据的图索引。
- **WITH ({storage_parameter = value} [, ...])**
  指定索引方法的存储参数。
  取值范围：如**表3-5**、**表3-6**所示。

GsIVFFLAT

表 3-5 向量索引类型 GsIVFFLAT

向量索引类型	函数名	描述
GsIVFFLAT	ivfflatbeginscan	读取索引根节点信息到内存，对于ivfflat读取元信息和各个聚簇中心点信息。
	ivfflatbuild	创建基于数据聚类的倒排索引。
	ivfflatbuildempty	反馈空索引。
	ivfflatbulkdelete	将所有索引中的死元组进行删除，并且更新每个聚簇的插入位置为第一个有空闲的页。
	ivfflatcostestimate	针对IVFFLAT扫描过程进行代价评估。
	ivfflatendscan	清空索引扫描结构体上的变量并释放。
	ivfflatgettuple	此函数提供执行器获得一条索引数据，在其中实现IVFFLAT核心检索步骤。
	ivfflatinsert	将新的元组插入到现有索引中，维护IVF索引。
	ivfflatoptions	为索引解析并验证reloptions数组。
	ivfflatrescan	重置索引扫描结构体上的变量。
	ivfflatvacuumcleanup	更新IVFFLAT索引统计信息。

GsDiskANN

表 3-6 向量索引类型 GsDiskANN

向量索引类型	函数名	描述
GsDiskANN	diskannbuild	创建基于Vamana图的向量索引。
	diskannbuildempty	反馈空索引。
	diskannoptions	为索引解析并验证reloptions数组。
	diskanninsert	将新的元组插入到现有索引中，维护diskann索引。
	diskannbeginscan	读取索引根节点信息到内存，加载PQ表中心点信息，指定的距离函数，关键结构创建。
	diskanngettuple	此函数提供执行器获得一条索引数据，在其中实现GSDISKANN核心检索步骤。

向量索引类型	函数名	描述
	diskannrescan	重置索引扫描结构体上的变量。
	diskannendscan	清空索引扫描结构体上的变量并释放。
	diskannbulkdelete	将所有索引中的死元组进行重置并且放入FSM中。
	diskannvacuumcleanup	更新GSDISKANN索引统计信息。
	diskanncostestimate	针对GSDISKANN扫描过程进行代价评估。
	diskannarrayconsistent	判断数组类型的查询条件是否与索引中的数据一致，用于决定是否需要重新检查，仅支持TRUE或者FALSE判断。
	diskannarrayextract	将数组类型的查询条件提取为多个元素，并设置“nkeys”和“null_flags”参数，用于后续的索引扫描。
	diskannqueryarrayextract	处理数组类型的查询条件，并根据数值比较策略设置搜索模式。
	diskannarraytriconsistent	判断数组类型查询条件与索引数据是否一致的函数。在diskannarrayconsistent基础上加入了对不确定的情况做“可能匹配”的判断，除TRUE或者FALSE，还支持MAYBE判断。

说明

- 创建索引时的内存需求超过限制会报错，内存需求的计算是统计当前数据量的大小，并根据统计数据量估算需要的内存大小，此提示值为统计值，大小可能会波动，建议根据实际情况在提示值基础上适当增大，将GUC参数maintenance_work_mem设置为合适大小。
- GsIVFFLAT索引支持的向量类型为floatvector和boolvector，而GsDiskANN索引仅支持向量类型为floatvector。

3.4 BM25 索引

实现了相似文本BM25索引访问方法，比如创建、维护、检索等。创建索引格式请参见《参考》中“SQL参考 > SQL语法 > C > CREATE INDEX”章节。BM25索引不支持UNIQUE、CONCURRENTLY。

- 重建索引请参见《参考》中“SQL参考 > SQL语法 > R > REINDEX”章节。
- 删除索引请参见《参考》中“SQL参考 > SQL语法 > D > DROP INDEX”章节。
- **GLOBAL**
  分区表GLOBAL索引不支持。
- **ASC**
  不支持。

- **CONCURRENTLY**

  不支持。
- **UNIQUE**

  不支持。
- **COLLATE collation**

  不支持。
- **DEDUPLICATION**

  不支持复合索引。
- **USING method**

  指定创建索引的方法。

  新增可选索引类型BM25：BM25索引，针对文本和文本数组数据的词倒排相似文本检索索引。
- **WITH ({storage_parameter = value} [, ...])**

  指定索引方法的存储参数。

  a. storage_parameter：数据存储格式，取值范围{"USTORE", "ASTORE"}，需要和原数据表保持一致。

  b. num_parallels：建索引并行度，取值范围1-64的整数，默认值为1。

  c. tokenize_dict：文本数据分词使用词典名，默认值pg_catalog.simple_cn_segmentation为系统默认内置中英文分词词典。具体词典使用和创建方法请参考**自定义Tokenweight分词词典**。

  d. token_hash_bucket_capacity：自定义存储token哈希表的大小，取值范围1000~100000000的整数，默认为10000000。

表 3-7 BM25 索引新增系统内部函数

函数名	描述
gs_bm25build	创建基于倒排词的BM25索引。
gs_bm25buildempty	反馈空索引。
gs_bm25options	为索引解析并验证reloptions数组。
gs_bm25insert	将新的元组插入到现有索引中，维护BM25索引。
gs_bm25beginscan	打开索引内部倒排词表容器，准备后续查询。
gs_bm25gettuple	此函数提供执行器获得一条索引数据，在其中实现BM25评分排序检索的核心步骤。
gs_bm25rescan	重置索引扫描结构体上的变量。
gs_bm25endscan	清空索引扫描结构体上的变量并释放。
gs_bm25bulkdelete	将所有索引中的死元组进行重置并且放入FSM中。
gs_bm25vacuumcleanup	更新BM25索引统计信息。
gs_bm25costestimate	针对BM25扫描过程进行代价评估。

表 3-8 BM25 索引相关新增系统函数

函数名	描述
gs_bm25_tokenize(text, cstring)	使用指定词典对输入文本进行分词操作，返回被分割的单词文本数组。
gs_bm25_inspect(text)	显示索引磁盘使用和内部词倒排信息等数据。
gs_bm25_distance_text(text, text)	返回bm25文档相似分数，只在使用BM25索引检索时有效。
gs_bm25_distance_textarr(text[], text[])	返回bm25文档相似分数，只在使用BM25索引检索时有效。

表 3-9 BM25 索引相关新增 GUC 参数

参数名	级别	描述
bm25_ranking_metric	会话级（session）	参数数值对应检索时相似文档的评分算法： 0：BM25_OKAPI (默认值), 1：BM25_ATIRE, 2：BM25_L, 3：BM25_PLUS, 4：TF_IDF。 取值范围: [0,127]，默认值0。设置为定义范围外的参数数值时使用默认BM25_OKAPI算法。
bm25_ncandidates	会话级（session）	预估索引召回文档数量，推荐使用默认值，过大或者过小的数值会使相似文档排序的索引检索时间变慢。 在查询语句混合where条件使用时，该值失效，数据库会自动估计合适的索引召回文档数量。 取值范围: [0,65535]，默认值128。

4 常见问题

创建表时，报错 dimensions for type vector cannot exceed 4096

答：创建表时，需指定向量维度，且不超过4096维。

```
gaussdb=#  CREATE TABLE t2 (a int, repr floatvector(100));
```

创建表时，报错 The "boolvector" type declaration for column must have length set

答：创建表时，需要指定向量维度。

```
gaussdb=#  CREATE TABLE t1 (a int, repr floatvector(2));
gaussdb=#  CREATE TABLE t2 (a int, repr boolvector(2));
```

若定义表类型含向量维度类型，插入数据非此类型数据，则报错。

答：插入数据需要符合数据类型。

```
gaussdb=#  INSERT INTO t1 VALUES(0, '[10.1, 0.5]');
```

插入数据时，若插入向量的字符串使用格式错误，则报错。

答：插入向量的字符数据需要符合格式。

```
gaussdb=#  INSERT INTO t1 VALUES(1, '[1,2]');
gaussdb=#  INSERT INTO t1 VALUES(2, '{1,2}');
```

若定义表类型含的向量维度和插入数据不符，则报错。

答：需要插入数据的维度和设定维度一致。

```
gaussdb=#  INSERT INTO t1 VALUES(1, '[1,0]');
gaussdb=#  INSERT INTO t2 VALUES(0, '[T,F]');
```

进行不同维度数据相似性检索时报错。

答：计算相似度的向量维度需要一致。

```
gaussdb=#  SELECT '[2,3,1]'<-> '[1,2,3]'::floatvector AS s;
```

创建索引时指定错误的索引参数数值，则报错。

答：创建索引时指定范围内的参数值。

```
gaussdb=#  CREATE TABLE sift1m (id int unique, repr floatvector(128));
gaussdb=#  CREATE INDEX diskann_l2_idx on sift1m using gsdiskann (repr L2) WITH (pq_nseg=128,
pq_nclus=16, queue_size=100, num_parallels=30, enable_pq=true, using_clustering_for_parallel=false);
```


创建索引时，若索引类型和属性类型不匹配，则报错。

答：索引类型和属性类型需要匹配，详细情况请参考向量索引章节。

```
gaussdb=#  CREATE INDEX h1 ON sift1m USING GSIVFFLAT(repr cosine) WITH (IVF_NLIST = 1024);
```

创建索引时，若向量维度过高，则报错。

答：高维度向量创建索引时，需启用量化压缩，并设置enable_vector_copy=false，详细情况请参见**向量索引GsDiskANN**章节。

```
gaussdb=#  CREATE TABLE sift2048 (id int unique, repr floatvector(2048));
gaussdb=#  CREATE INDEX diskann_l2_idx on sift2048 using gsdiskann (repr L2) WITH (pq_nseg=128,
pq_nclus=16, queue_size=100, num_parallels=30, enable_pq=true, using_clustering_for_parallel=false,
build_with_quantized_vector=true, subgraph_count=1, enable_vector_copy=false);
```

在线创建/在线重建索引时，若同时并发执行 DDL（删表、删索引、重建索引等操作）和 DML（插入、删除等操作），则可能触发死锁并报错。

```
gaussdb=# CREATE TABLE t1 (a int, repr floatvector(2));
gaussdb=# INSERT INTO t1 VALUES(0, '[10.1, 0.5]');
gaussdb=# CREATE INDEX CONCURRENTLY t1ann on t1 USING GsDiskANN(repr L2);
gaussdb=# DROP TABLE t1;
gaussdb=# UPDATE TABLE t1 set repr = '[10.2, 1.0]' where a = 0;
```

在线创建/在线重建索引时，若自动扩展分区表插入新数据或更新导致扩区，则报错。

```
gaussdb=# CREATE TABLE t1 (a int, repr floatvector(2)) partition by list(a) AUTOMATIC (partition p0 values
(1));;
gaussdb=# INSERT INTO t1 VALUES(0, '[10.1, 0.5]');
gaussdb=# CREATE INDEX CONCURRENTLY t1ann on t1 USING GsDiskANN(repr L2);
```

大量删除数据后进行索引查询时，返回数据不足，小于 limit 的值。

答：考虑检查索引健康度，看是否需要重建索引。

```
gaussdb=#  SELECT * FROM gs_diskann_inspect('diskann_sift1m_l2');
```

在创建索引时，数组字段元素数量超过 128 个，则报 WARNING。

答：数组字段元素数量超过128个，索引会正常创建，但不会保证性能。

```
gaussdb=#  CREATE TABLE test_table (id int, a bigint,repr floatvector(3), arr129 bigint[]);
gaussdb=#  INSERT INTO test_table (id, a, repr, arr129) VALUES
(1,99,'{0.1,0.2,0.3}',ARRAY[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,6
8,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,1
03,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,
129]);
gaussdb=#  CREATE INDEX hybrid_idx_t0 on test_table using gsdiskann(repr L2, a, arr129)
with(pq_nseg=3,pq_nclus=16,queue_size=128,num_parallels=30,subgraph_count=5);
WARNING:  Array column has more than 128 elements, which may affect performance.
```

在创建索引时，数组字段列超过 2 列，则报错。

```
gaussdb=#  CREATE TABLE test_table( id INT, a BIGINT, repr floatvector(16),e INT[], f BIGINT[], j
SMALLINT[]);
gaussdb=#  CREATE INDEX hybrid_idx2 on test_table using gsdiskann(repr L2, id, e, f, j)
with(num_parallels=30,subgraph_count=5);
ERROR:  The index contains more than two array columns, which is not allowed.
```


在创建向标混合索引时，指定的标量数组字段的数组维度大于 1 维，则报错。

```
gaussdb=#  CREATE TABLE t1(id int, repr floatvector(128), a int, b int, c int[][]);
gaussdb=#  CREATE INDEX hybrid_idx_t1 on t1 using gsdiskann(repr L2, a, b, c)
with(pq_nseg=128,pq_nclus=16,queue_size=128,num_parallels=30,subgraph_count=5);
ERROR:  The index parameter is invalid. The following problems may occur:1.The scalar column data type is restricted.2.Only support one vector type and must be in the first column.3.To support scalar filtering, the value of subgraph_count must be greater than 0.
```

在创建向标混合索引时，指定的标量数组字段的元素类型为 bytea 类型（或其他非法类型），则报错。

```
gaussdb=#  CREATE TABLE t1(id int, repr floatvector(128), a int, b int, c bytea[]);
gaussdb=#  CREATE INDEX hybrid_idx_t1 on t1 using gsdiskann(repr L2, a, b, c)
with(pq_nseg=128,pq_nclus=16,queue_size=128,num_parallels=30,subgraph_count=5);
ERROR:  The index parameter is invalid. The following problems may occur:1.The scalar column data type is restricted.2.Only support one vector type and must be in the first column.3.To support scalar filtering, the value of subgraph_count must be greater than 0.
```

当用户进行混合查询时，如果标量字段属于混合索引的标量字段集，且为数组类型，但是对数组元素使用 LIKE 后匹配（或其他不支持的操作符）。

答：正常执行，但是对该语句执行explain语句会发现该LIKE过滤条件不被混合索引处理。

```
gaussdb=#  CREATE TABLE t1(id int, repr floatvector(3), a int, b int, c varchar(128)[]);
gaussdb=#  CREATE INDEX hybrid_idx_t1 on t1 using gsdiskann(repr L2,  a, b, c)
with(pq_nseg=3,pq_nclus=16,queue_size=128,num_parallels=30,subgraph_count=5);
gaussdb=#  INSERT INTO t1 (id, repr, a, b, c) VALUES (1001,'[1.1,2.2,3.3]',100,1,ARRAY['vectordb_test',
'other_value']::varchar(128)[]),
(1002, '[2.2,3.3,4.4]', 101, 1, ARRAY['vectordb_test']::varchar(128)[]),
(1003, '[3.3,4.4,5.5]', 100, 2, ARRAY['vectordb_test']::varchar(128)[]),
(1004, '[4.4,5.5,6.6]', 100, 1, ARRAY['other_value']::varchar(128)[]),
(1005, '[5.5,6.6,7.7]', 101, 2, ARRAY['other_value']::varchar(128)[]),
(1006, '[6.6,7.7,8.8]', 100, 1, ARRAY['test_value']::varchar(128)[]),
(1007, '[7.7,8.8,9.9]', 100, 1, ARRAY['abc']::varchar(128)[]),
(1008, '[8.8,9.9,1.1]', 100, 1, ARRAY['nobody_test']::varchar(128)[]);
gaussdb=#  explain SELECT /*+ indexscan(t1 hybrid_idx_t1) */ id FROM t1 WHERE a=100 and b < 2 and c[1]
LIKE '%vectordb%' order by repr <-> '[1,2,3]' limit 10;
```

用户进行混合查询时，标量数组字段使用了特定函数。

答：正常执行，但是该数组字段的过滤条件不被混合索引处理。

```
gaussdb=#  CREATE TABLE t1(id int, repr floatvector(3), a int, b int, c int[]);
gaussdb=#  CREATE INDEX hybrid_idx_t1 on t1 using gsdiskann(repr L2,  a, b, c)
with(pq_nseg=3,pq_nclus=16,queue_size=128,num_parallels=30,subgraph_count=5);
gaussdb=#  INSERT INTO t1 (id, repr, a, b, c) VALUES (1, '[1.1, 2.2,
3.3]',100,1,ARRAY[1,2,3,4,5,6,7,8,9,10,11] ),
(2, '[2.0, 3.0, 4.0]', 99, 1, ARRAY[1,2,3,4,5,6,7,8,9,10]),
(3, '[3.1, 4.2, 5.3]', 100, 2, ARRAY[1,2,3,4,5,6,7,8,9,10,11]),
(4, '[4.0, 5.0, 6.0]', 100, 1, ARRAY[1,2,3,4,5,6,7,8,9,10]),
(5, '[5.5, 6.6, 7.7]', 101, 0, ARRAY[1,2,3,4,5,6,7,8,9,10,11,12]),
(6, '[6.1, 7.2, 8.3]', 100, 3, ARRAY[1,2,3,4,5,6,7,8,9,10,11,12]),
(7, '[7.7, 8.8, 9.9]', 100, 1, ARRAY[1,2,3,4,5,6,7,8,9,10]),
(8, '[8.0, 9.0, 10.0]', 101, 1, ARRAY[1,2,3,4,5,6,7,8,9,10,11,12,13]);
gaussdb=#  explain SELECT /*+ indexscan(t1 hybrid_idx_t1) */ id FROM t1 WHERE a=100 and b < 2 and
array_length(c, 1) > 10 order by repr <-> '[1,2,3]' limit 10;
```

在创建向标混合索引时，CHAR 和 VARCHAR 参数大于 256，则报错。

```
gaussdb=#  CREATE TABLE t1(id int, repr floatvector(128), col varchar(257)[10]);
gaussdb=#  CREATE INDEX hybrid_idx_t1 on t1 using gsdiskann(repr L2, col) with(subgraph_count=5);
ERROR:  The VARCHAR column exceeds the maximum allowed length of 256 characters.
```


分区表在执行文档相似性查询时，如果存在任意分区是不可用的，则报错。

```
gaussdb=# ALTER INDEX test_bm25idx MODIFY PARTITION p1 UNUSABLE;
ALTER INDEX
gaussdb=# SELECT /*+ indexscan(test test_bm25idx) */id, text, text ### '客户' as relevance_score FROM test
ORDER BY relevance_score DESC LIMIT 10;
ERROR:  Some partitions' indexes are unusable, this is not currently supported.
```

