第九届PostgreSQL中国技术大会

2019 PostgreSQL Conference China

开源驱动 自主研发

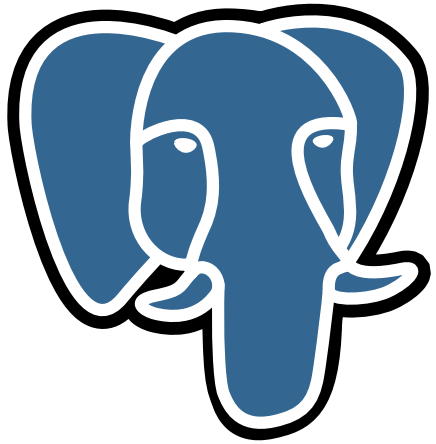
主办：PostgreSQL中文社区

协办：ITPUB

2019年11月29日-30日

北京维景国际大酒店

PostgreSQL高维向量检索索引插件介绍



姓名：方概
公司：蚂蚁金服

蚂蚁金服 ANT FINANCIAL

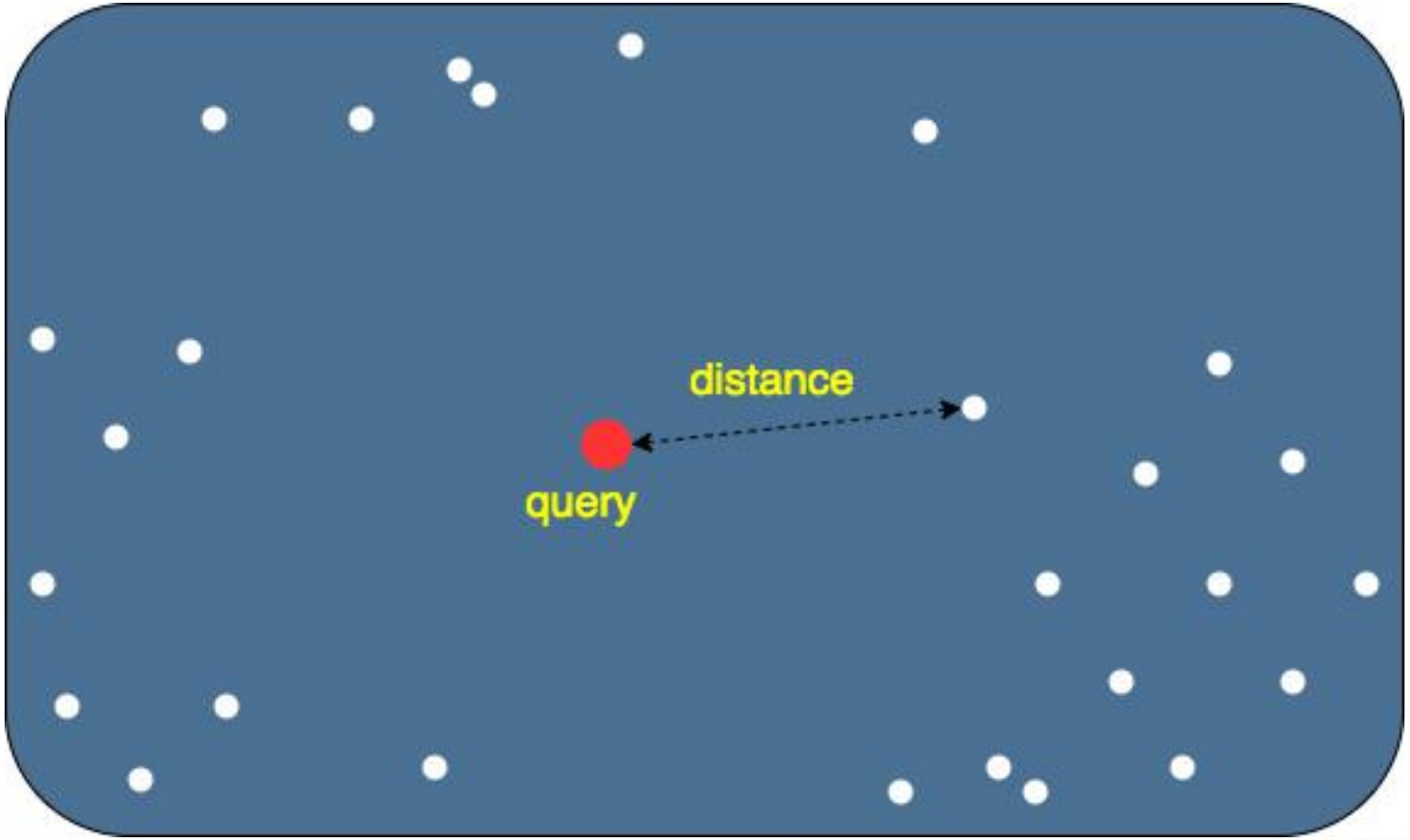
目录

01 背景

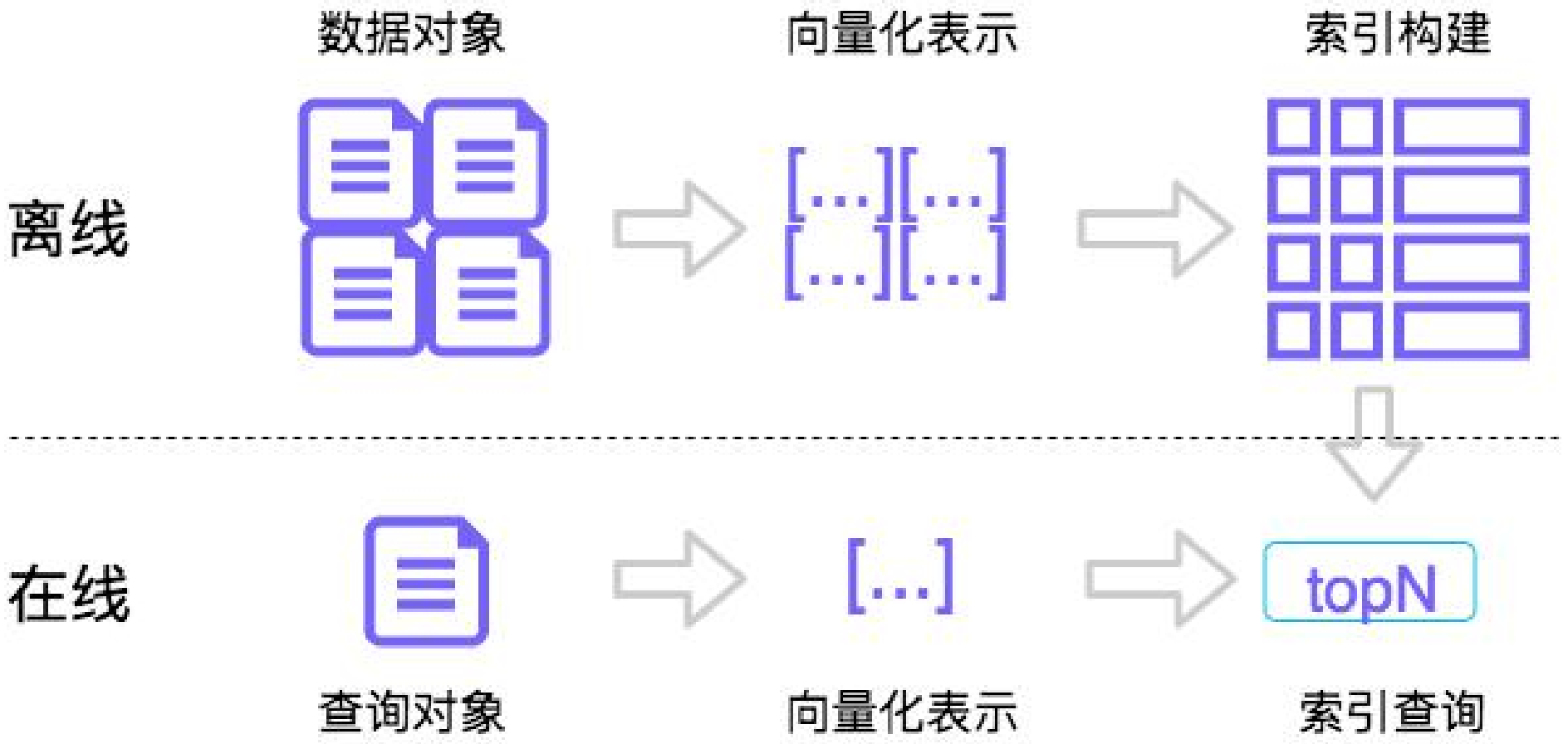
问题，解决方案

蚂蚁金服
ANT FINANCIAL

近似最近邻检索
(ANN,Approximate Nearest Neighbor Search)



ANN流程



向量检索场景

为什么不直接使用开源算法库

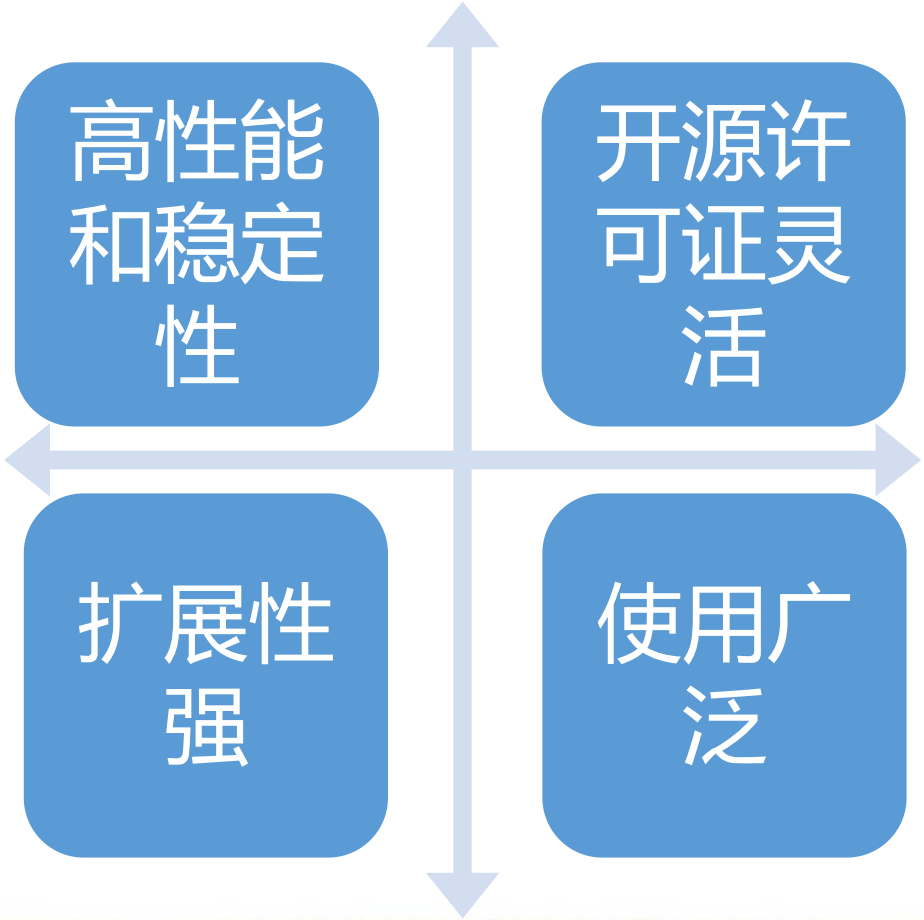
- 开源算法库：faiss， SPTAG, annoy, nmslib, falconn

问题

- 组合查询
- 分布式和高可用
- 数据迁移

PostgreSQL(简称PG)优势

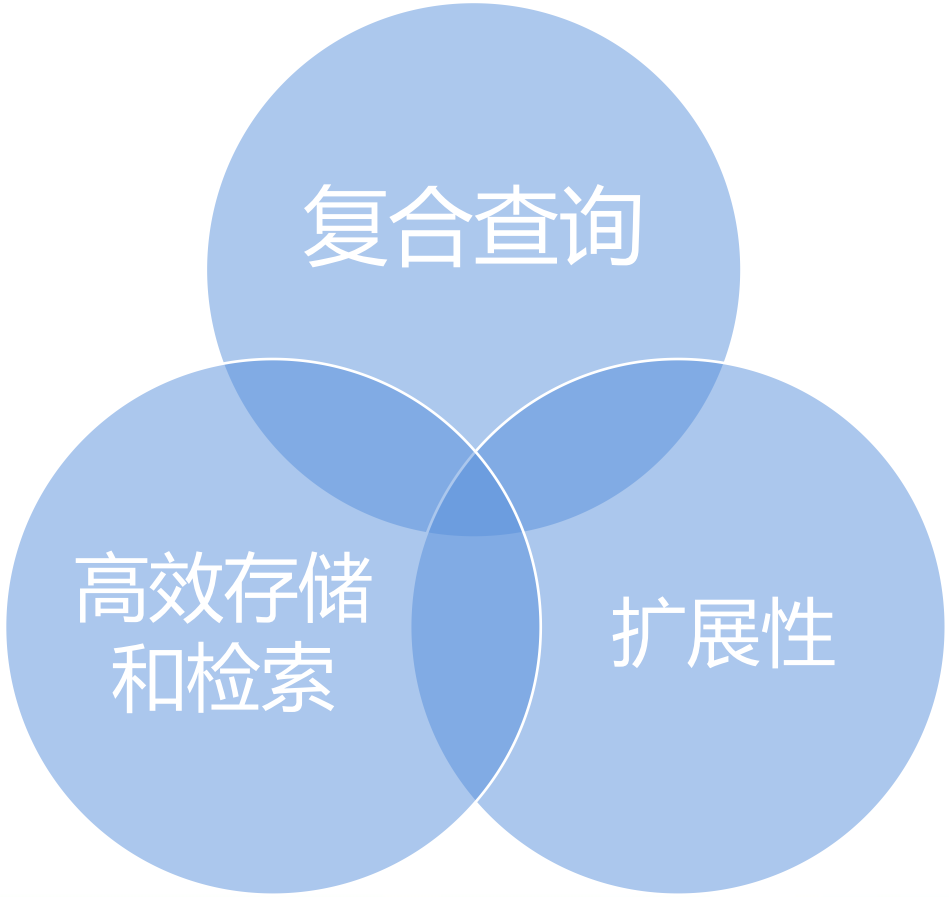
The World's Most Advanced Open Source Relational Database



PG现有向量索引插件

索引插件	实现方式	特点	限制
imgsmlr	基于Gist索引	内部提供图像特征提取等方法	只能用于16维的向量检索场景，不适用于大规模的工业应用
cube	基于Gist索引	支持向量的聚类、向量的距离计算	仅支持100维以下，测试性能不佳
freedy	基于上层SQL扩展函数	支持丰富的ANN算法	大规模数据场景性能无法接受

基于PG的高维向量检索插件
(Pase - PG ANN search extension)



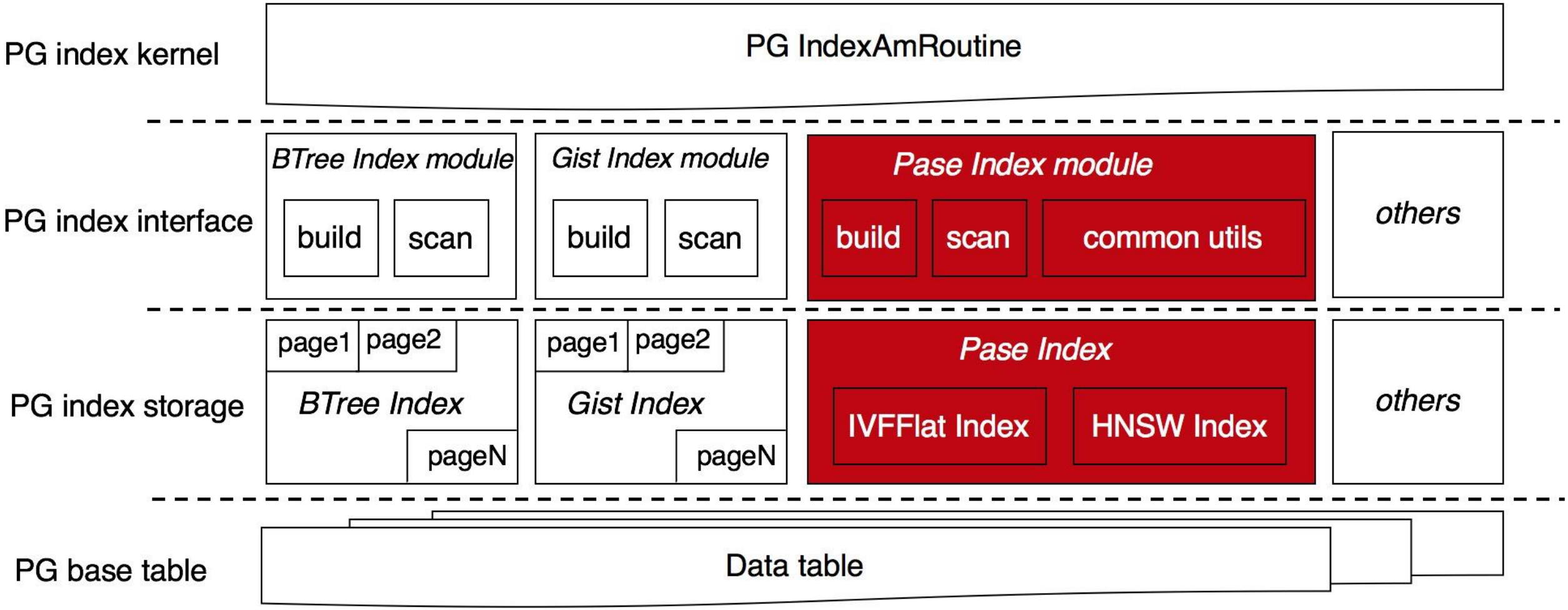
一种可行的通用方案，赋予PG大规模高维向量检索能力。

02 索引插件实现

索引结构，实现算法和索引扩展

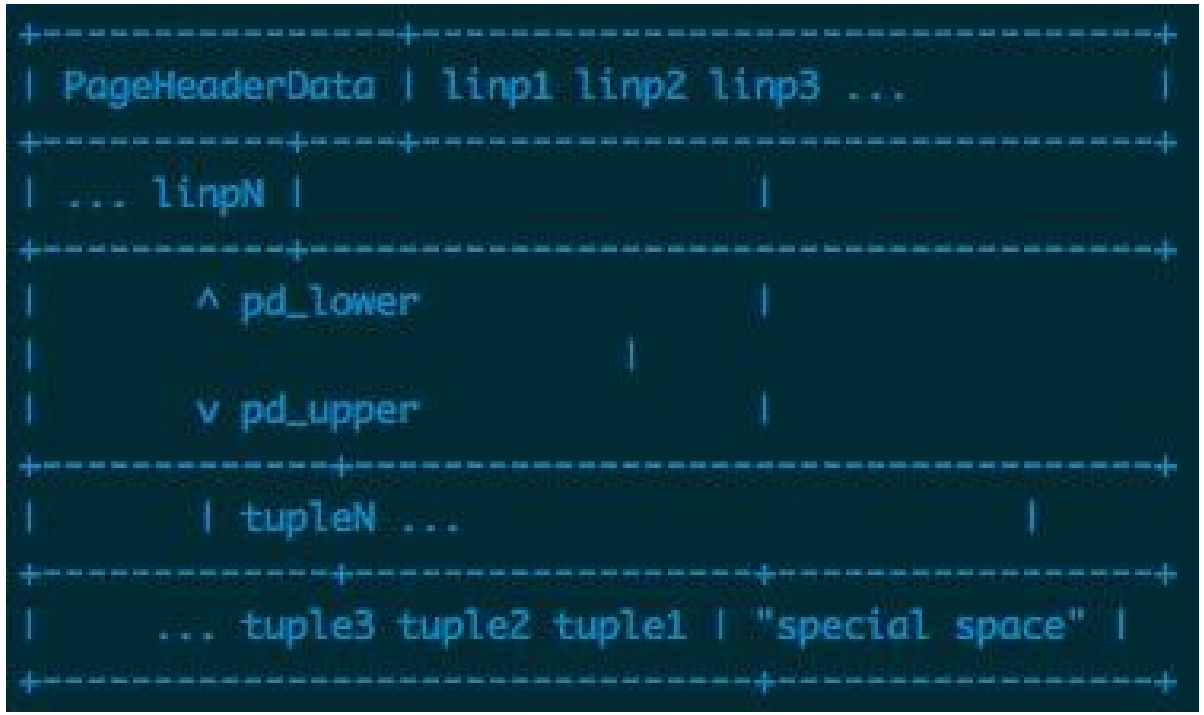
蚂蚁金服
ANT FINANCIAL

PG索引结构抽象


PG index kernel
PG IndexAmRoutine
PG index interface
BTree Index module
build
scan
Gist Index module
build
scan
Pase Index module
build
scan
common utils
others
PG index storage
page1
page2
BTree Index
pageN
page1
page2
Gist Index
pageN
Pase Index
IVFFlat Index
HNSW Index
others
PG base table
Data table

PG索引底层存储结构

底层存储基于PG定义的基础结构：PAGE



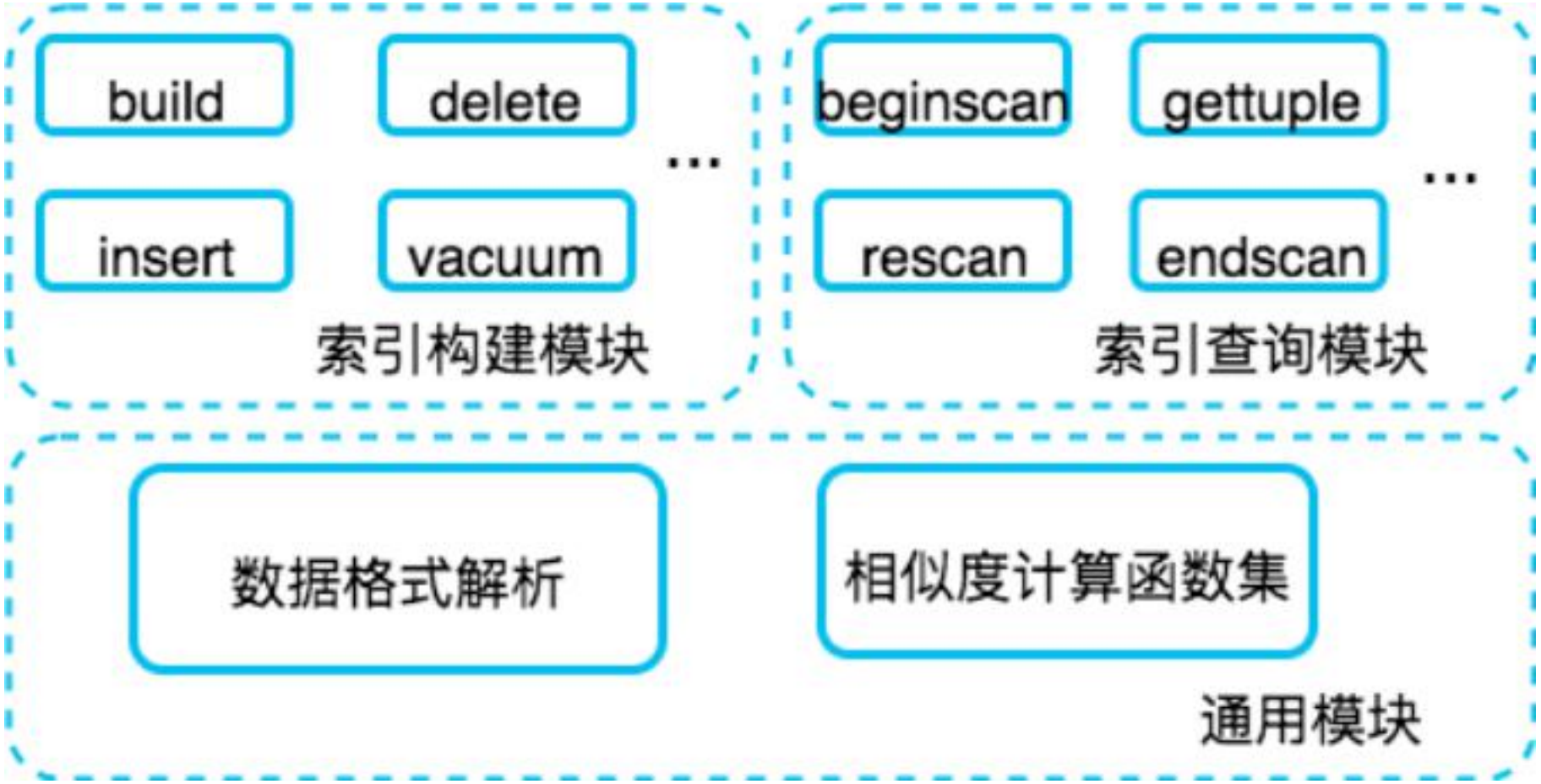
PageHeaderData - PG定义的数据结构，用于Page的管理。
special space – 部分内容由用户自定义使用。
User define space - 用户使用，内部的存储单元为DataTuple结构。

索引接口函数

IndexAmRoutine结构

接口名称	作用
ambuild	创建一个新索引
ambuildempty	构建一个空索引
aminsert	向现有索引插入一个新元组
ambulkdelete	从索引中删除元组，批量删除
amvacuumcleanup	索引的清理和整合
amcostestimate	估计一次索引扫描的开销
amoptions	分析和验证一个索引的 reloptions 数组
ambeginscan	开始一个新扫描
amrescan	重启一个扫描
amgettuple	获取下一个元组，向给出的方向移动
amgetbitmap	获取所有可用的元组
amendscan	结束扫描并释放资源

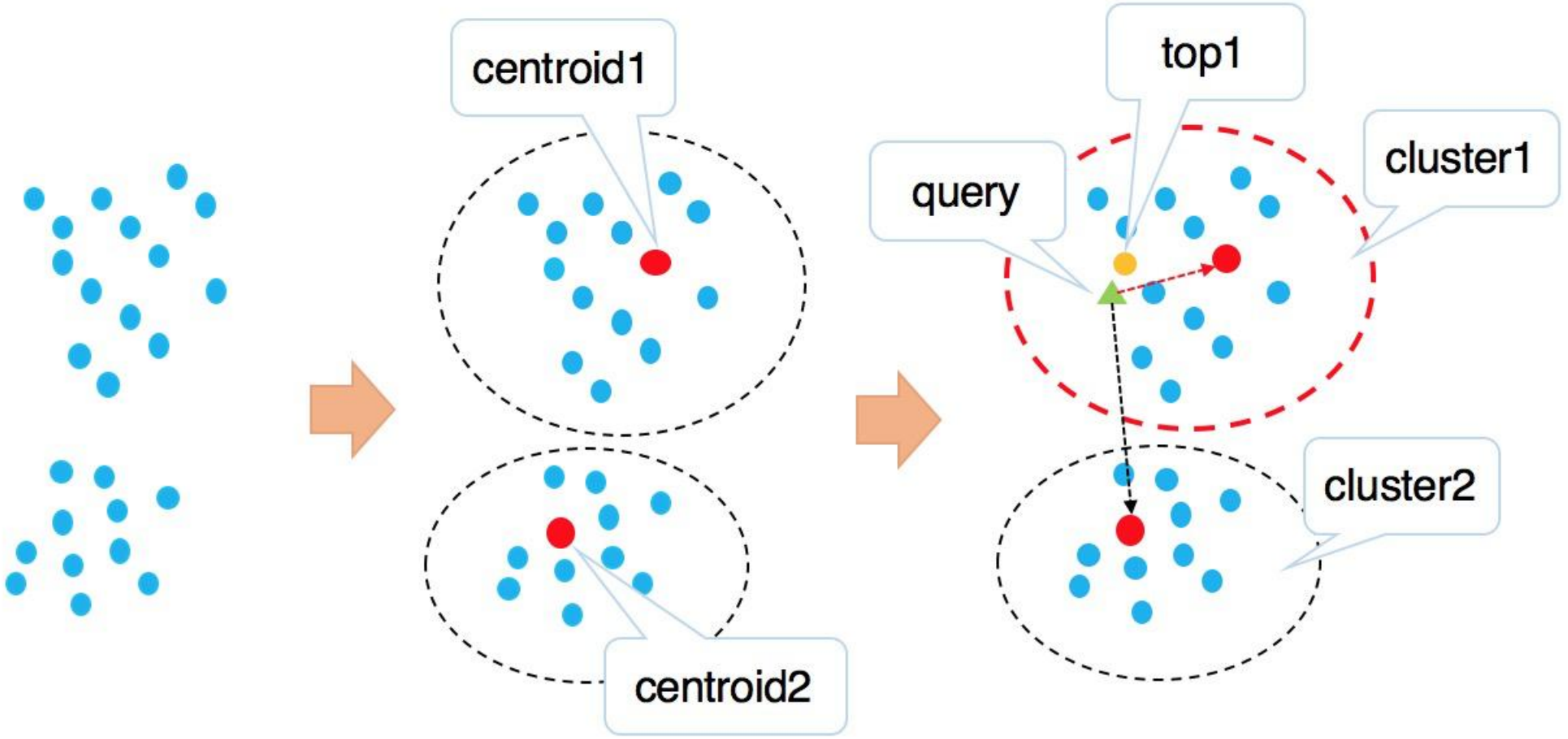
插件c代码组成


build
delete
...
insert
vacuum
索引构建模块
beginscan
gettuple
...
rescan
endscan
索引查询模块
数据格式解析
相似度计算函数集
通用模块

算法选择

基于树的	基于哈希的	基于量化的	基于图的
• KDTree • RTree • ...	• LSH • ...	• IVFFlat • IVFPQ • IVFADC • IMI • HC • ...	• HNSW • NSG • SSG • ...

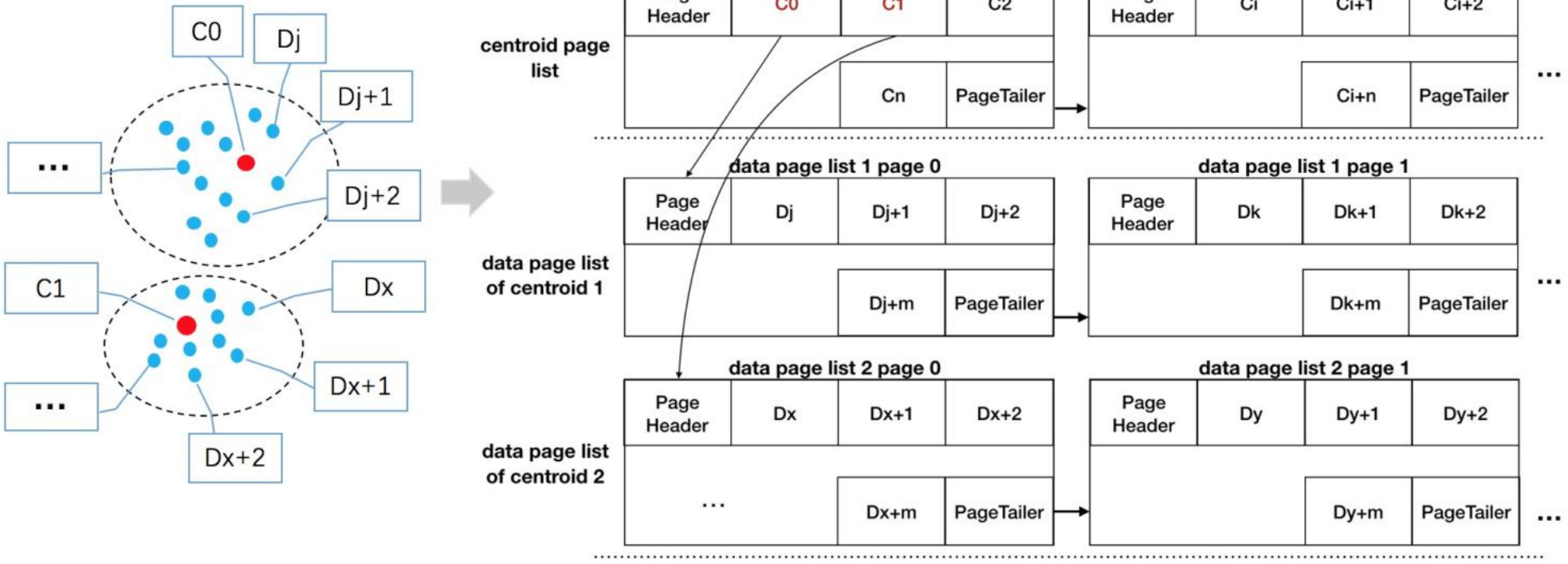
IVFFlat算法



适用场景：

适合于召回精度要求高，但对查询耗时要求不严格（100ms级别）的场景

IVFFlat-存储结构

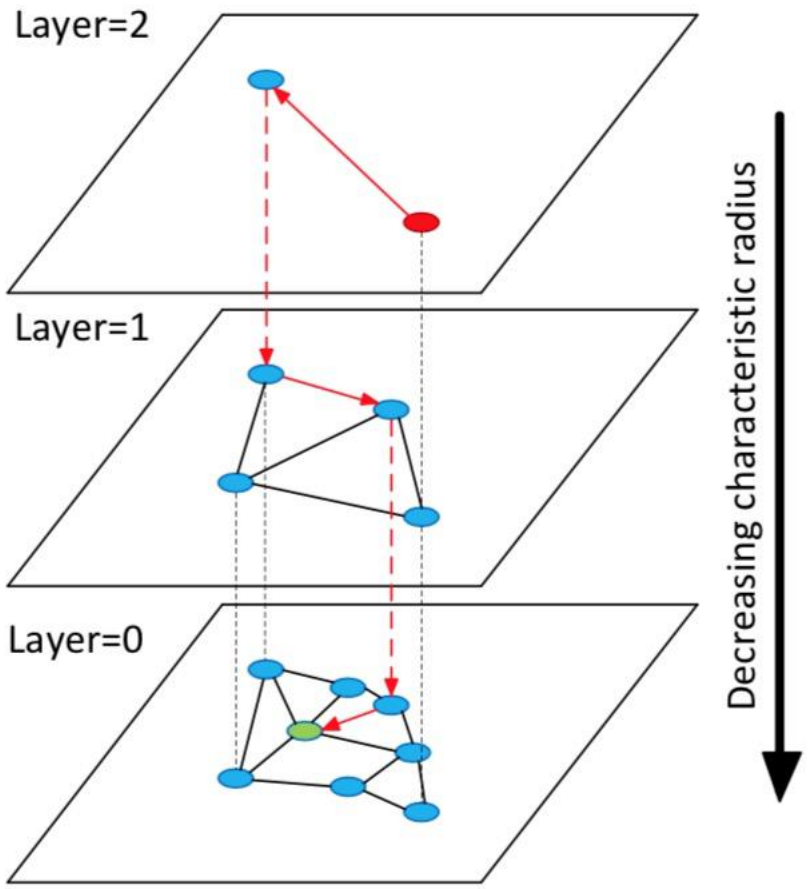


优化-page连续存储

1. 一次申请N个page
2. vacuum: 重新申请page

HNSW算法

分层可导航小世界图
（Hierarchical Navigable Small World, HNSW）



适用场景：

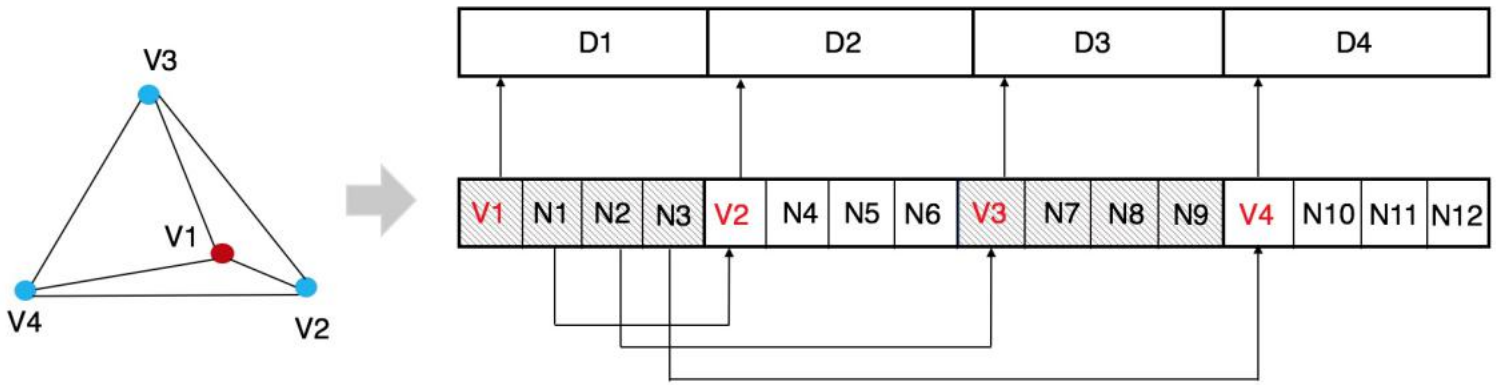
超大规模向量数据集（千万级别），且对查询延时有严格要求(10ms级别)的场景

HNSW算法-存储结构

- meta page
  存储索引的meta信息：入口点、最后一个data page id，以及初始化配置参数。

- data page
  存储原始向量数据，每个单元内部存储原始的向量数据、节点所属层次。形成page链。

- neighbor page
  存储邻接关系。



自定义数据类型

输入输出函数声明(.sql)

```
CREATE FUNCTION pase_in(cstring)
RETURNS pase
AS 'MODULE_PATHNAME'
LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;

CREATE FUNCTION pase_out(pase)
RETURNS cstring
AS 'MODULE_PATHNAME'
LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;
```

输入输出函数实现(.c)

```
PG_FUNCTION_INFO_V1(pase_in);

Datum
pase_in(PG_FUNCTION_ARGS) {
  PASE *result;
  result = (PASE *)palloc0(size);
  ...
  PG_RETURN_POINTER(result);
}
```

自定义操作符

```
CREATE FUNCTION g_pase_distance(float4[], pase)
RETURNS float4
AS 'MODULE_PATHNAME'
LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;

CREATE OPERATOR <?> (
  LEFTARG = float4[],
  RIGHTARG = pase,
  PROCEDURE = g_pase_distance,
  COMMUTATOR = '<?>'
);
```

自定义数据类型plm(.sql)

```
CREATE TYPE pase (
  INTERNALLENGTH = variable,
  INPUT = pase_in,
  OUTPUT = pase_out,
  RECEIVE = pase_recv,
  SEND = pase_send,
  ALIGNMENT = float
);
```

跟IndexAmRoutine的连接

SQL

```
CREATE FUNCTION pase_hnsw(internal)
RETURNS index_am_handler
AS 'MODULE_PATHNAME'
LANGUAGE C;

CREATE ACCESS METHOD pase_hnsw
TYPE INDEX
HANDLER pase_hnsw;
```

C代码

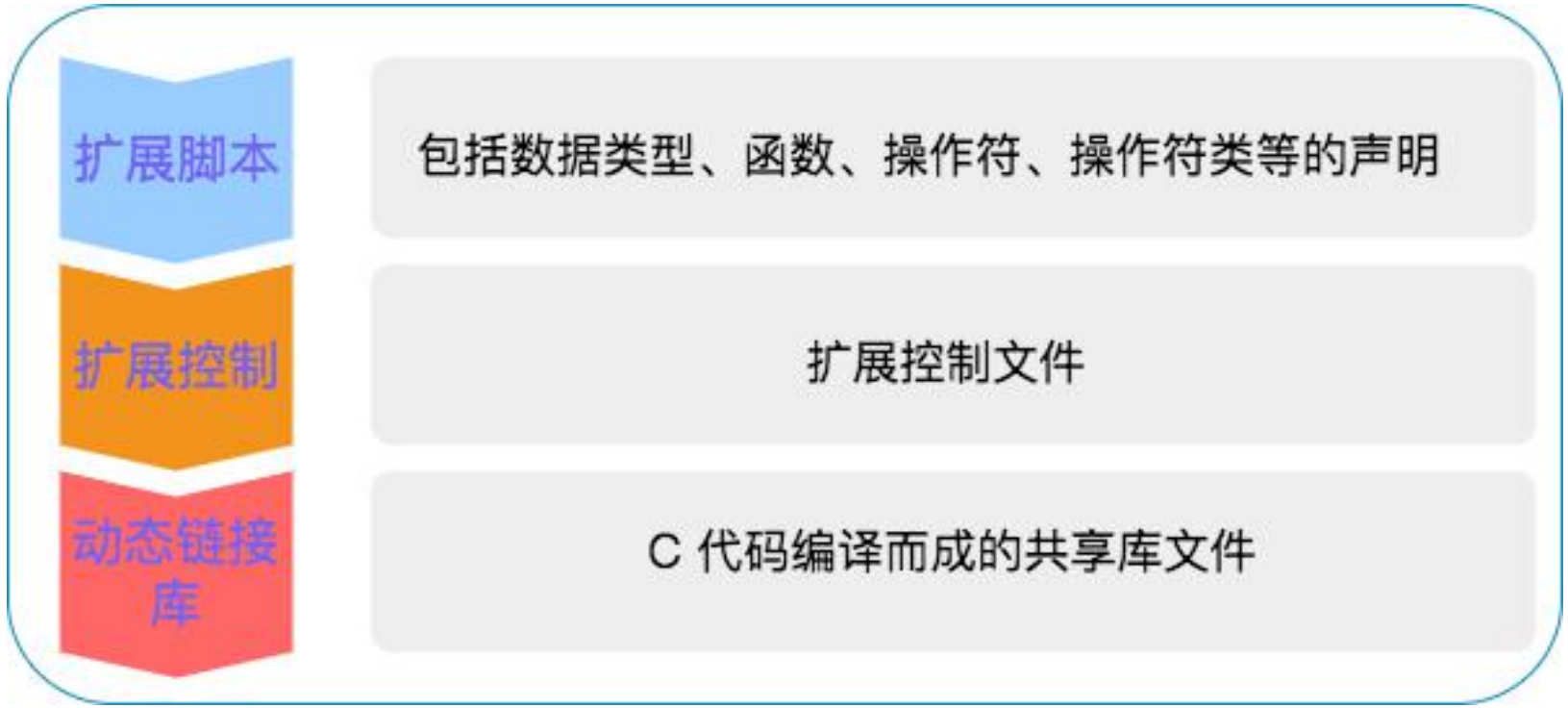
```
PG_FUNCTION_INFO_V1(pase_hnsw);

Datum
pase_hnsw(PG_FUNCTION_ARGS) {
  IndexAmRoutine *amroutine = makeNode(IndexAmRoutine);

  ...
  amroutine->ambuild = hnsw_build;
  amroutine->ambuildempty = hnsw_buildempty;
  amroutine->aminsert = hnsw_insert;
  amroutine->amcostestimate = hnsw_costestimate;
  amroutine->amoptions = hnsw_options;
  amroutine->ambeginscan = hnsw_beginscan;
  amroutine->amrescan = hnsw_rescan;
  amroutine->amgettuple = hnsw_gettuple;
  amroutine->amendscan = hnsw_endscan;
  ...

  PG_RETURN_POINTER(amroutine);
}
```

最终成果：Extension(扩展)



参考：

$PG_SRC/contrib/cube
$PG_SRC/contrib/bloom

03 示例和性能

使用示例和性能数据

蚂蚁金服
ANT FINANCIAL

索引构建-HNSW

构建语法

```
-- create table
CREATE TABLE vectors (
  id serial,
  vector float4[]
);
```

```
-- insert data
…
```

```
CREATE INDEX v_hnsw_idx ON vectors
USING
  pase_hnsw(vector)
WITH
  (dim = 768, base_nb_num = 12, ef_build = 50, ef_search = 100, base64_encoded = 0);
```

构建性能

数据量-维度	索引构建时间
100w-256维	1小时~几小时
2500w-512维	几小时~十几小时

索引查询-HNSW

查询语法

```
SET enable_seqscan=off;
SET enable_indexscan=on;
SELECT id, vector <?> '...'::pase as distance
FROM vectors_hnsw_test
ORDER BY
vector <?> '...'::pase
 ASC LIMIT 10;
```

... 为逗号分隔的256个浮点数，这里省略

为了表达按照向量相似度返回的语义，Pase的查询功能通过order by语句来实现。
order by支持通过索引扫描来加速(amgettuple接口)。

索引构建-IVFFlat

构建语法

```
CREATE INDEX v_ivfflat_idx ON vectors
USING
  pase_ivfflat(vector)
WITH
  (clustering_type = 0, distance_type = 0, dimension = 256, base64_encoded = 1, clustering_params =
   "/data/centroid_path/centroids.v45.20190802");
```

构建性能

中心点数量：1000

数据量-维度	索引构建耗时
100w 256维	约7分钟
2500w 512维	约48分钟

性能-索引构建耗时和索引大小

测试环境

64X Intel(R) Xeon(R) CPU E5-2682 v4 250GB RAM, 1.8TB disk, centos 7

相同的PG配置，单线程

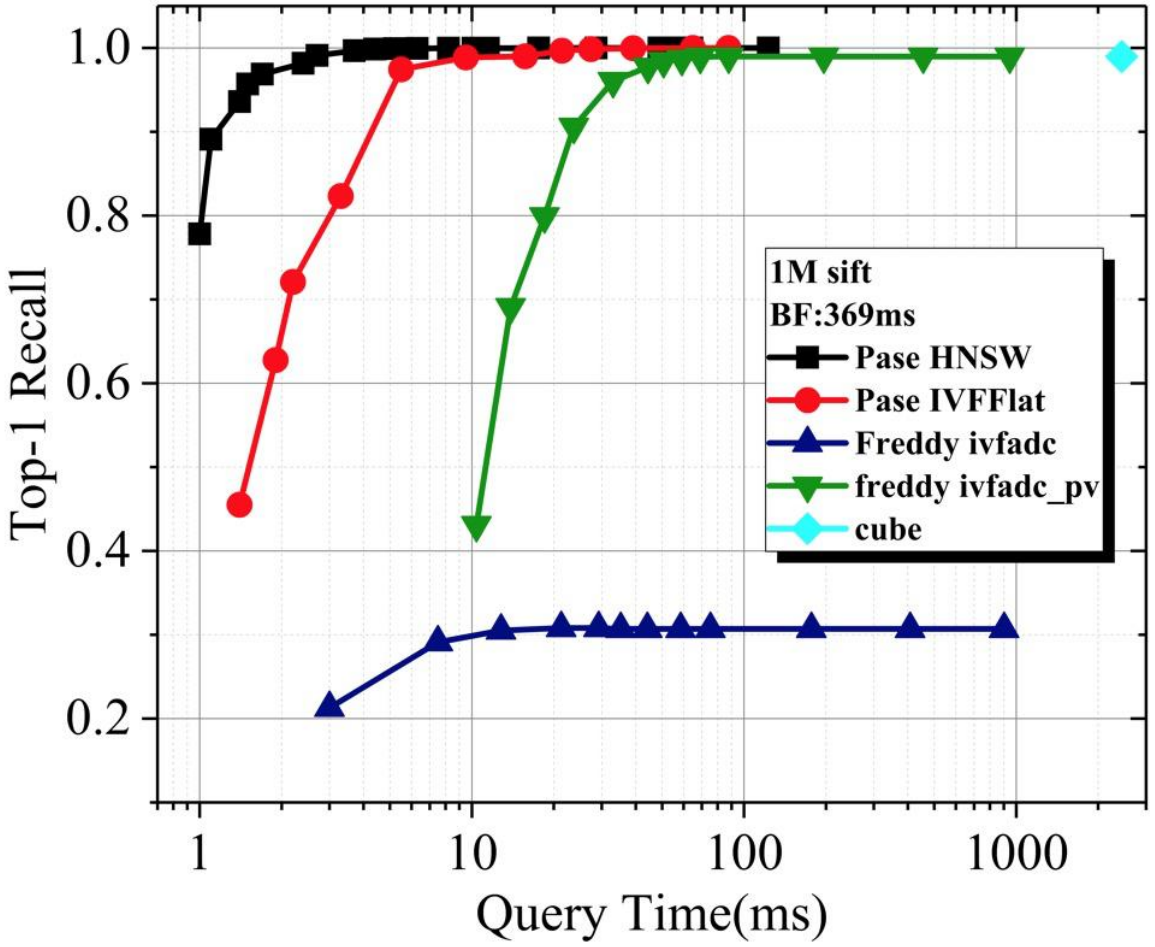
sift1M(128维)数据集

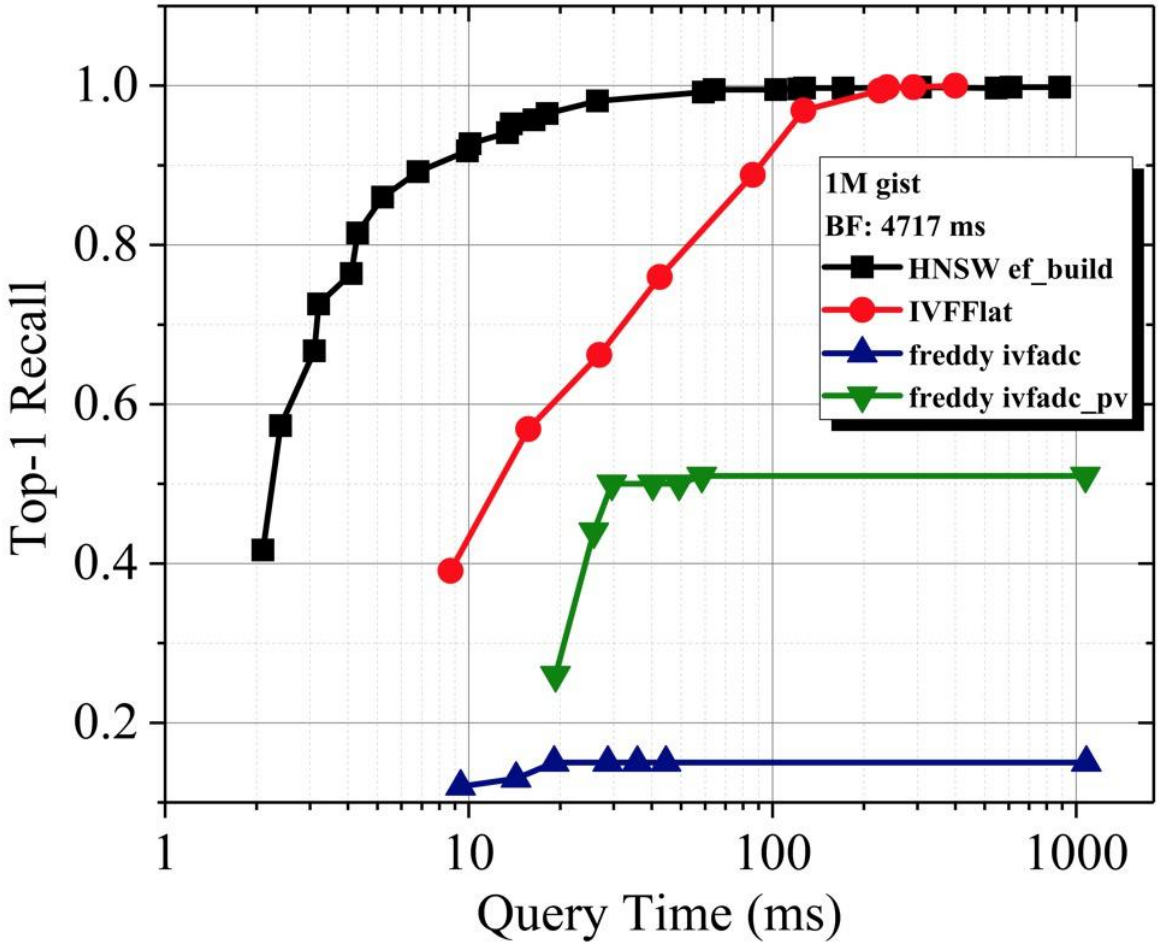
Extension	Build time (second)	Table size (MB)	index size (MB)
cube	314	1116	2877
freedy	1020	558	101
IVFFlat	255	558	525
HNSW	4942	558	8333

gist1M(960维)数据集

Extension	Build time (second)	Table size (MB)	index size (MB)
freedy	4388	3813	116
IVFFlat	372	3806	3912
HNSW	20875	3806	11718

性能-耗时和准确率


1.0
0.8
0.6
0.4
0.2
Top-1 Recall
1
10
100
1000
Query Time(ms)
1M sift
BF:369ms
Pase HNSW
Pase IVFFlat
Freddy ivfadc
freddy ivfadc_pv
cube

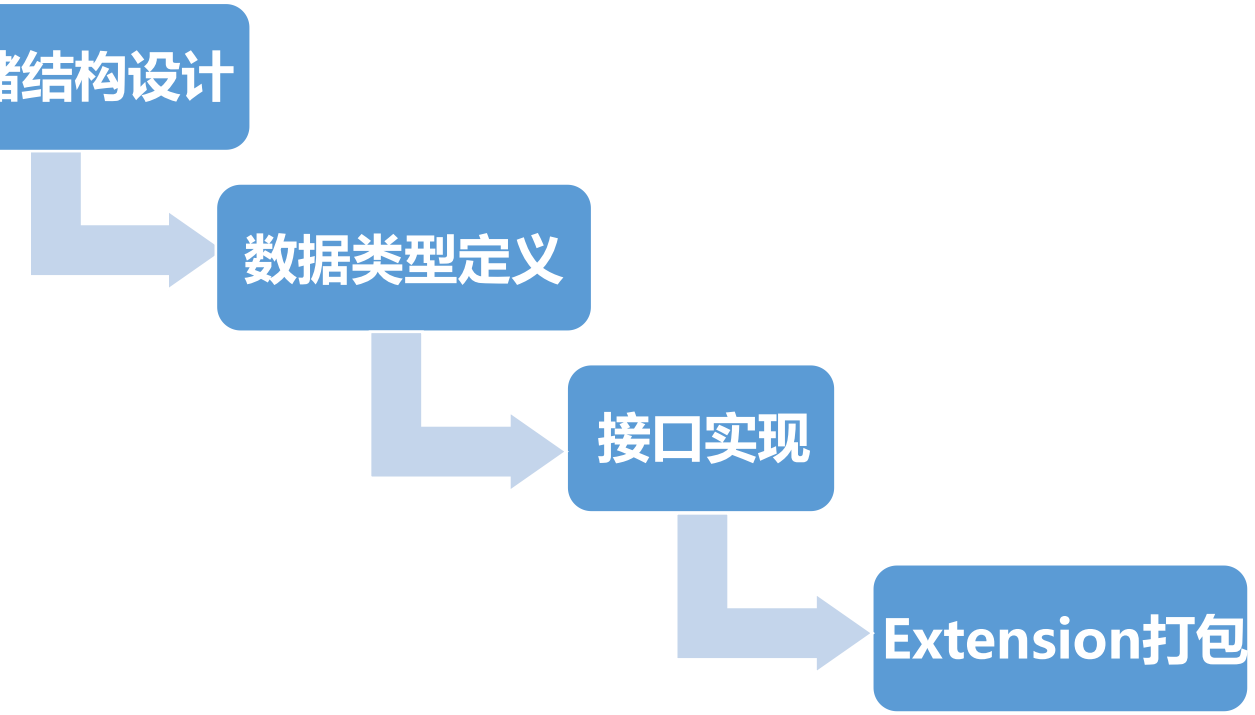

1.0
0.8
0.6
0.4
0.2
Top-1 Recall
1
10
100
1000
Query Time (ms)
1M gist
BF: 4717 ms
HNSW ef_build
IVFFlat
freddy ivfadc
freddy ivfadc_pv

04 PG新索引扩展步骤

蚂蚁金服
ANT FINANCIAL

PG扩展新索引步骤抽象

- 1.根据算法原理设计索引的page结构和组织关系
- 2.定义新数据类型和新距离计算函数
- 3.实现IndexAmRoutine里的部分接口
  build, insert
  beginscan, rescan, endscan, getdatatuple
  delete, vacuum
- 4.Extention打包
  脚本文件，控制文件，动态链接库



05 展望

- 实现优化：存储设计优化，并行build，查询性能优化。
- 算法优化和扩充。HNSW算法有很多可优化点。引入优秀的新算法。
- 已经进入阿里云pg版本中，外部客户正在进行测试。
- 开源流程进行中。

蚂蚁金服
ANT FINANCIAL

蚂蚁金服智能搜索团队 欢迎交流

THANKS

蚂蚁金服
ANT FINANCIAL

参考

1. FAISS https://github.com/facebookresearch/faiss
2. SPTAG https://github.com/microsoft/SPTAG
3. imgsmlr https://github.com/postgrespro/imgsmlr
4. cube https://www.postgresql.org/docs/11/cube.html
5. freedy FREDDY: Fast Word Embeddings in Database Systems
6. ANN https://en.wikipedia.org/wiki/Nearest_neighbor_search#Approximate_nearest_neighbor
7. kdtree https://en.wikipedia.org/wiki/K-d_tree
8. IMI The Inverted Multi-Index
9. NSG https://github.com/ZJULearning/nsg#reference
10. SSG https://github.com/ZJULearning/SSG
11. HNSW Yu.A.Malkov, D.A.Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs.