# 向量索引

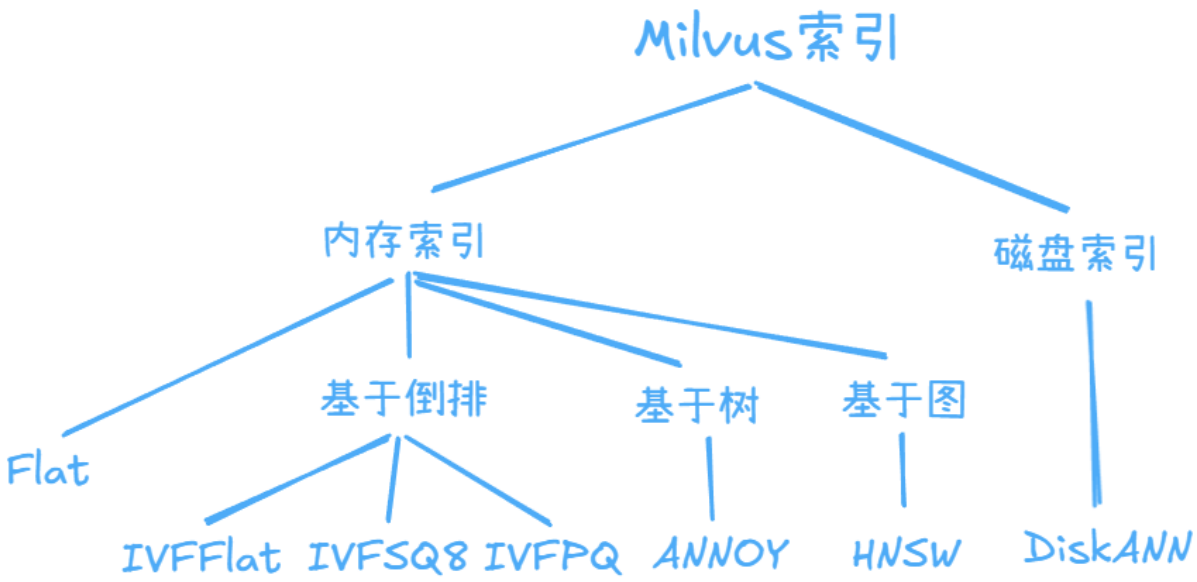
LLM领域大热的两个概念：RAG和Agent，都需要用到向量数据库，比如RAG中需要把知识库向量化之后存在向量数据库中，Agent中也需要将memory存储在外部存储器中。Milvus支持多种向量索引算法，这些算法各自基于不同的原理，旨在优化存储效率和搜索效率，满足不同应用场景的需求。

本文将深入探讨Milvus所支持的几种主要向量索引的原理，包括它们的工作机制、优势以及适用场景，以帮助读者更好地理解和选择适合自己需求的索引方案。

## ANNS

在处理高维数据时，最近邻搜索（NNS, Nearest Neighbor Search）是一个常见且重要的任务。NNS旨在通过给定的查询向量，快速找到数据集中最相似的若干个向量。这在图像检索、推荐系统、语音识别等应用中具有广泛的需求。然而，随着数据规模的增大，精确的最近邻检索通常会变得非常耗时和资源密集。因此，近似最近邻搜索（ANNS, Approximate Nearest Neighbor Search）应运而生。

ANNS的核心思想是在可接受的精度范围内，牺牲部分准确性，换取更高的检索效率。相比于精确检索，ANNS只需要找到目标向量的近似邻居，而不是完全精确的邻居，从而在大规模数据集上大幅提升查询速度。Milvus 支持的向量索引类型大多采用ANNS算法，常见的索引类型的划分如下图所示：

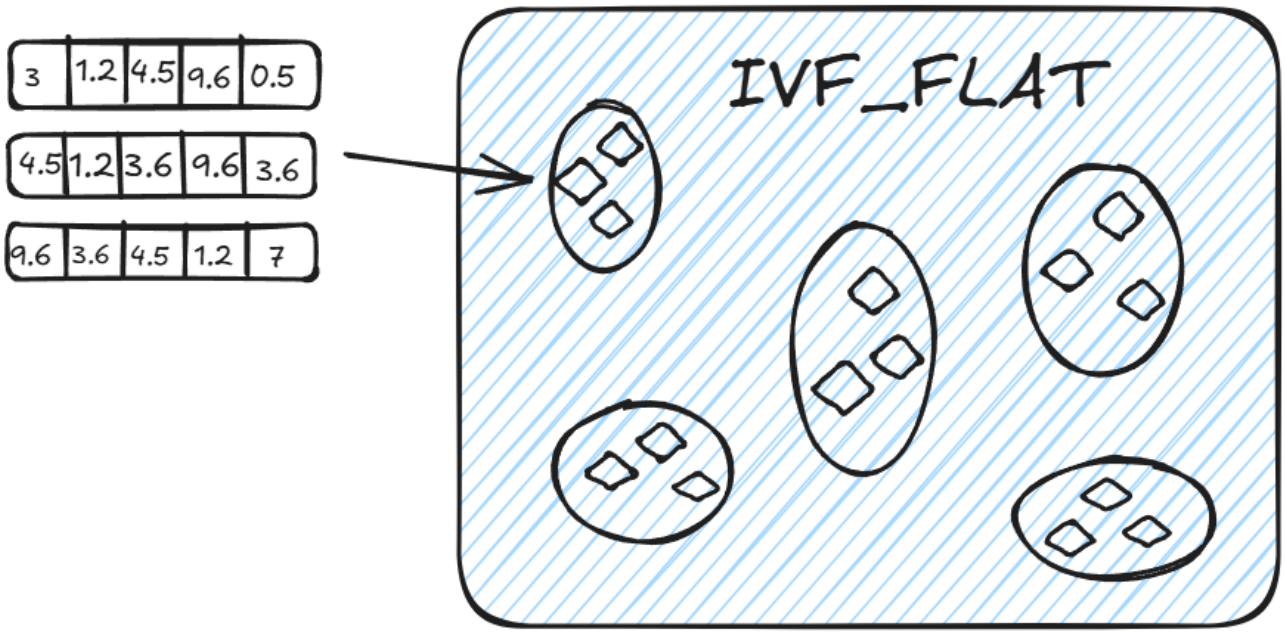



## FLAT

这是最简单的索引方式，进行暴力搜索（brute-force），可以保证精确度，但效率低，尤其在数据量大时。适合场景：在小型、百万级数据集上寻求完全精确的搜索结果。

# IVF_FLAT

**IVF_FLAT** 是一种基于倒排的索引方法，广泛用于在大规模数据集上实现高效的近似最近邻搜索。它适用于在精度和查询速度之间寻求平衡的场景。IVF_FLAT本身并没有进行量化操作，因此在精度和存储开销上相对保守，但能够提供较快的搜索速度。




## 核心原理

1. **聚类**：IVF_FLAT通过聚类算法（如k-means）将高维空间中的向量划分为多个子空间（簇）。每个簇包含一组相似的向量，并且每个簇会有一个代表向量，通常是簇的中心点。
2. **倒排索引**：为每个簇创建倒排索引。每个向量会被映射到它所属的簇，这样在查询时，系统只需关注与查询向量相似的簇，而不需要搜索整个高维空间，从而显著降低搜索的时间复杂度。
3. **查询处理**：
   - 查询时，IVF_FLAT首先将查询向量分配到距离最近的簇中心（即子空间）。
   - 然后在该簇内执行精确的线性搜索，从而查找与查询向量相似的向量。
   - 为了优化查询，IVF_FLAT使用一个参数 `nprobe` 来控制搜索的簇数。 `nprobe` 控制搜索时考虑的簇的数量，从而平衡查询精度和查询速度：
     - 增大 `nprobe` 可以搜索更多簇，返回更多候选向量，提高结果的精确度，但查询时间也会增加。
     - 减少 `nprobe` 可以缩小搜索范围，降低计算时间，查询速度更快，但可能会牺牲一些精度。

4. **降低搜索成本**：由于IVF_FLAT通过划分子空间来限制搜索范围，它能够显著减少传统线性搜索所带来的高维数据中的计算开销，从而提高查询效率。与传统的暴力搜索方法相比，IVF_FLAT的时间复杂度大大降低，尤其适合在大规模数据集上使用。

### 适用场景

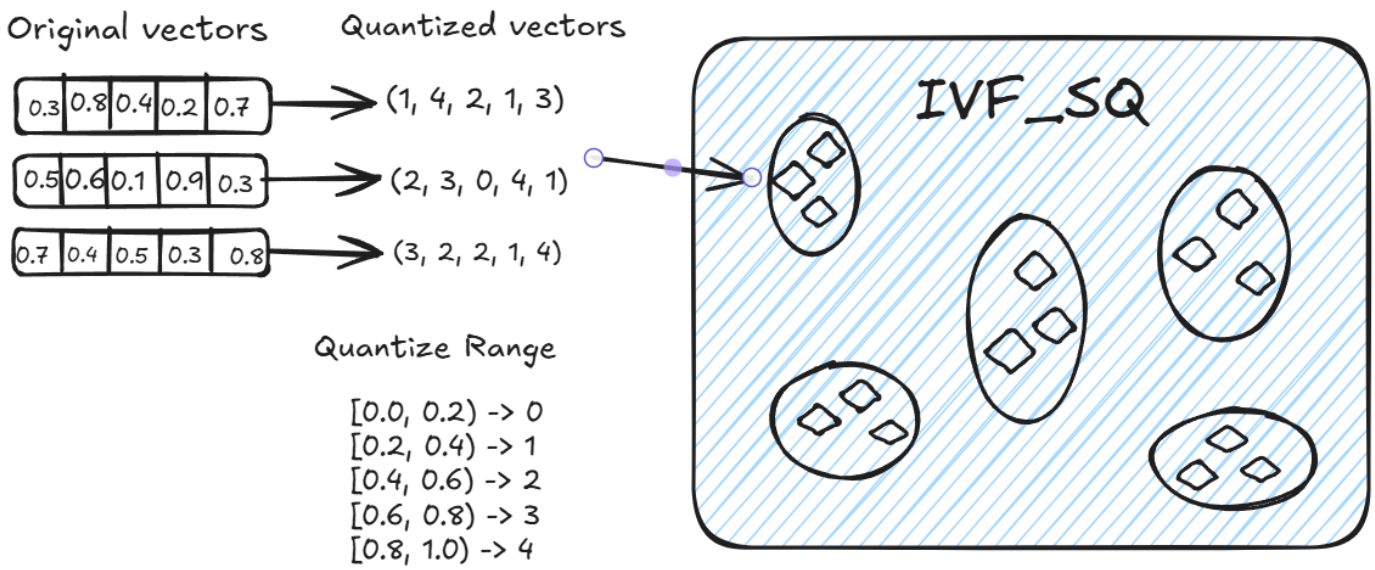
IVF_FLAT适用于需要平衡精度和查询速度的场景，尤其是在大规模、高维数据集上，可以有效减少查询时间。它适合那些要求较高精度但能容忍一定查询延迟的应用。

## IVF_SQ8

**IVF_SQ8** 是在 IVF_FLAT 基础上增加了量化步骤的一种索引方法，其核心思想与 IVF_FLAT 类似，但通过量化技术将存储和计算资源的消耗大大降低，尤其在磁盘、内存、CPU 和 GPU 资源的使用上节省了 70%-75%。IVF_SQ8通过**标量量化**（Scalar Quantization）将每个维度的 4 字节浮点数表示压缩为 1 字节整数表示。

### 核心原理

1. **标量量化**：IVF_SQ8 通过标量量化将每个向量的每个维度从 4 字节（通常是浮点数）压缩为 1 字节。量化的过程是将原始的浮点数值映射到一个较小的整数范围。例如，假设一个维度的原始值范围是 [0.0, 1.0]，通过量化后，该维度的数值会被压缩为整数值，这样可以显著节省存储空间并加速计算。
2. **Quantized Vectors**：量化后的向量使用整数（如 uint8）来表示每个维度的值。通过量化，向量的存储空间大大减少，同时查询时计算量也降低。量化后的整数表示会根据原始值的分布划分为若干个区间。
3. **倒排索引与聚类**：与 IVF_FLAT 类似，IVF_SQ8 使用聚类算法（如 k-means）将高维空间中的向量划分为多个簇。每个簇内的向量都通过量化后的表示存储和检索。查询时，系统会将查询向量分配到与其最接近的簇中心，然后在该簇内执行快速的线性搜索。



# IVF_PQ

**IVF_PQ** 是一种结合了倒排文件和乘积量化（Product Quantization, PQ）的高效索引方法，旨在加速大规模高维数据集的检索过程。它主要用于高维向量的近似最近邻搜索，通过将向量空间划分为更小的子空间并进行量化，显著降低了存储开销和计算复杂度。

1. 倒排文件

倒排文件是一种高效的索引结构，用于存储和检索向量。在IVF_PQ中，数据集中的每个向量被分配到一个或多个倒排表中，每个表包含了对应向量的标识符。查询时，我们首先在倒排文件中找到候选的向量集合，从而大大减少了搜索空间。倒排文件特别适合于高维空间，因为它允许我们仅搜索与查询向量相似的部分数据，而不是遍历整个数据集。

2. 乘积量化（PQ）

乘积量化是一种将高维向量压缩为低维表示的技术。它通过将向量划分为多个子空间，并对每个子空间进行独立的量化，生成一个代码本（codebook）。这样，原始的高维向量可以由多个子空间的量化表示组合而成，从而降低存储需求并加速检索。

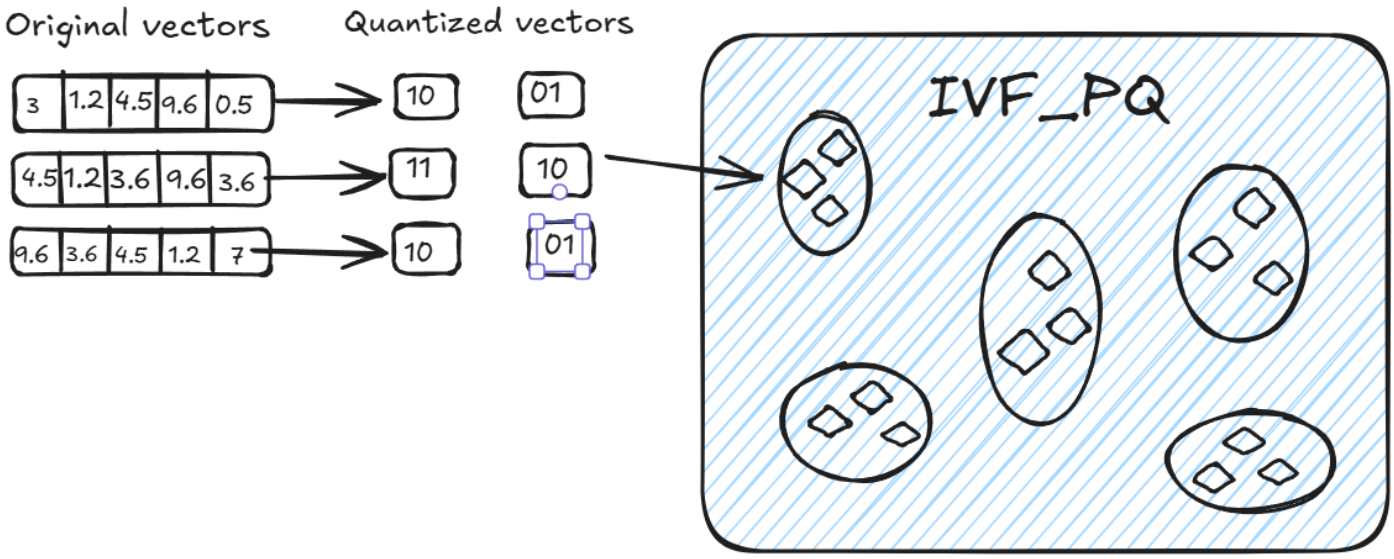
在IVF_PQ中，乘积量化应用于IVF的聚类过程。每个簇的中心点会被进一步量化，原始的查询向量和数据向量在计算距离时，不是直接与每个簇中心进行计算，而是与每个子空间的量化中心进行计算。这种方法不仅降低了存储开销，还减少了计算距离时的运算量。

3. IVF_PQ的结合

IVF_PQ将倒排文件和乘积量化结合在一起，利用两者的优势来加速高维向量检索。具体流程如下：

1. **量化与聚类**：首先，数据集中的每个向量会被分为多个子空间，每个子空间进行乘积量化。接着，通过倒排文件将数据按簇组织。
2. **查询流程**：
    - 查询时，首先根据查询向量找到相应的倒排表（即查询向量属于哪个簇）。
    - 然后，在该簇内，使用乘积量化后的代码本来进行相似度计算，找到与查询向量最相似的向量。

这样，通过倒排文件限制搜索范围，并通过乘积量化精简计算过程，IVF_PQ大大提高了大规模数据集上相似向量检索的效率。




# HNSW

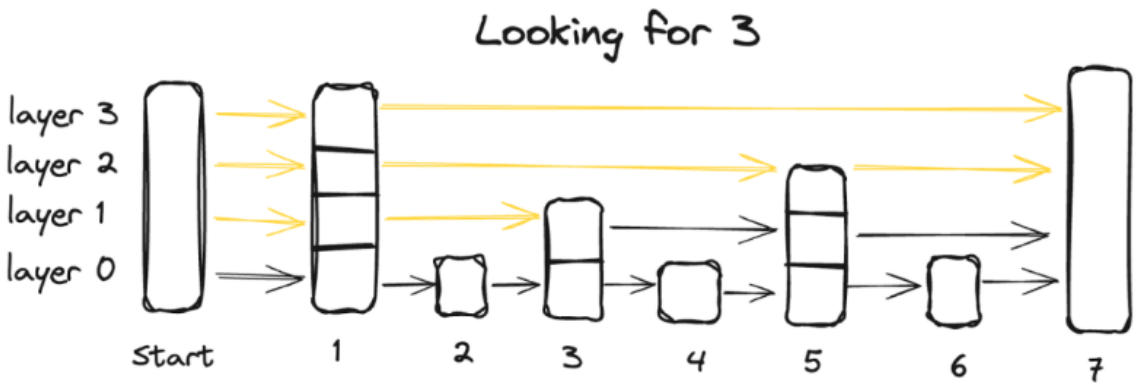
**HNSW**（Hierarchical Navigable Small World Graph）是一种基于图的索引算法，采用分层结构和小世界图理论，旨在高效地进行近似最近邻搜索。它通过构建一个多层次的图结构，其中每一层的节点连接关系不同，逐层精细化，从而提高高维数据集的搜索效率。

1. 图的结构

HNSW的图结构结合了两种技术：**跳表**（Skip List）和**可导航小世界**（NSW）图。

**跳表**特点：

- **多层链表**：跳表的底层是一个完整的有序链表，存储所有元素。上层链表是下层链表的“抽象版”，包含部分元素，随着层数增加变得更加稀疏。
- **逐层查找**：查询时，从最上层开始查找，如果当前层无法找到目标元素，则跳到下一层继续查找，直到最底层。




**可导航小世界（NSW）**特点：

- **邻接列表**：每个节点连接若干相似节点，称为邻接节点。每个节点都保存一个邻接列表。
- **遍历过程**：从随机选定的入口节点开始，通过图的边逐步找到最接近查询向量的节点。

2. HNSW的工作原理

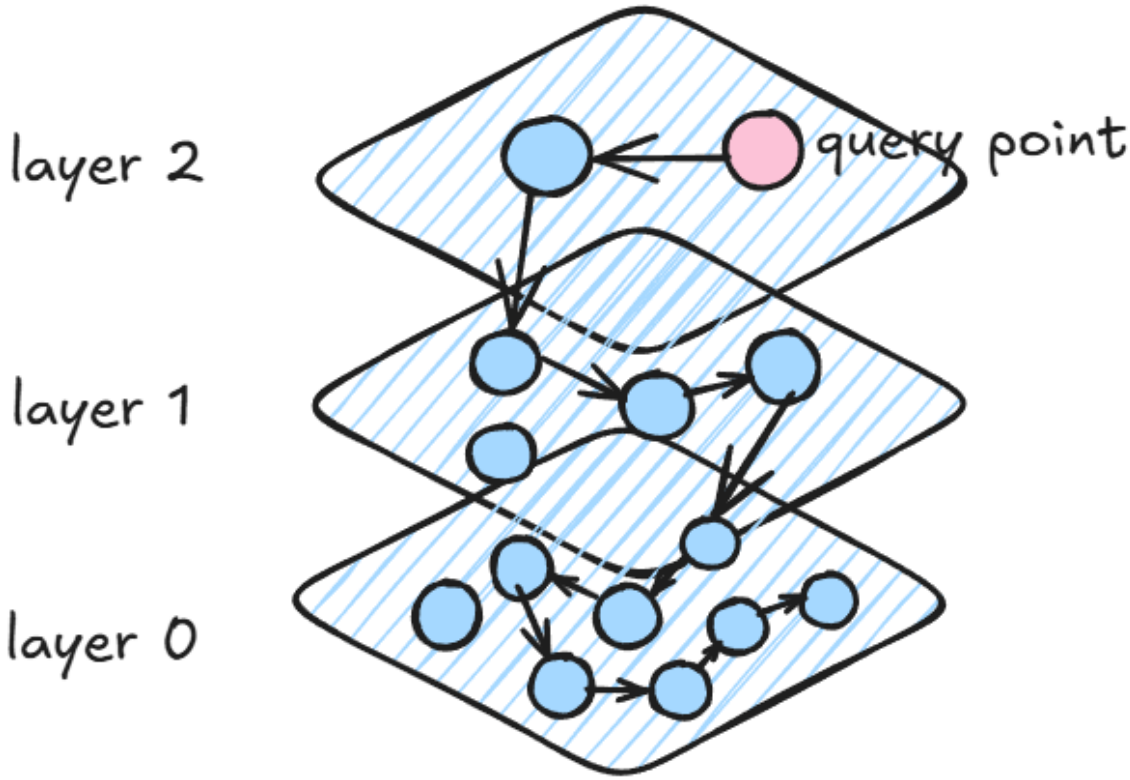
HNSW将跳表的层次化结构与NSW的小世界理论结合起来，形成了一个高效的近似最近邻搜索算法。其工作分为两个主要阶段：索引构建和查询过程。

索引构建

- **图的层次结构**：HNSW构建一个多层图，每一层代表不同的搜索精度和速度。最上层图的节点较少，提供粗粒度的搜索；而底层节点则提供更精细的搜索，逐层提升搜索精度。
- **连接邻居**：每个新加入的节点会选择若干个近邻节点进行连接，从而形成一个局部的小世界结构。通过选择性地建立邻接关系，确保了图的稀疏性和高效搜索。

查询过程

- **逐层搜索**：查询从最上层图开始，逐层向下进行。每一层会根据相似度从当前节点跳到相邻节点，逐步逼近目标位置。此时，查询会通过图中的边，利用跳表的方式，逐步接近查询向量。
- **局部优化**：在最底层，HNSW通过局部搜索策略，遍历当前节点的邻接节点，找到最接近查询向量的结果。




# DiskANN

DiskANN是一种基于磁盘的高性能向量近邻搜索算法，旨在解决大规模向量数据检索中的内存消耗问题。通过将轻量级的索引结构置于内存中，而将海量的原始数据和构建好的图结构存放在磁盘上，DiskANN能够在保持高召回率和低时延的同时，大幅减少对内存资源的依赖。

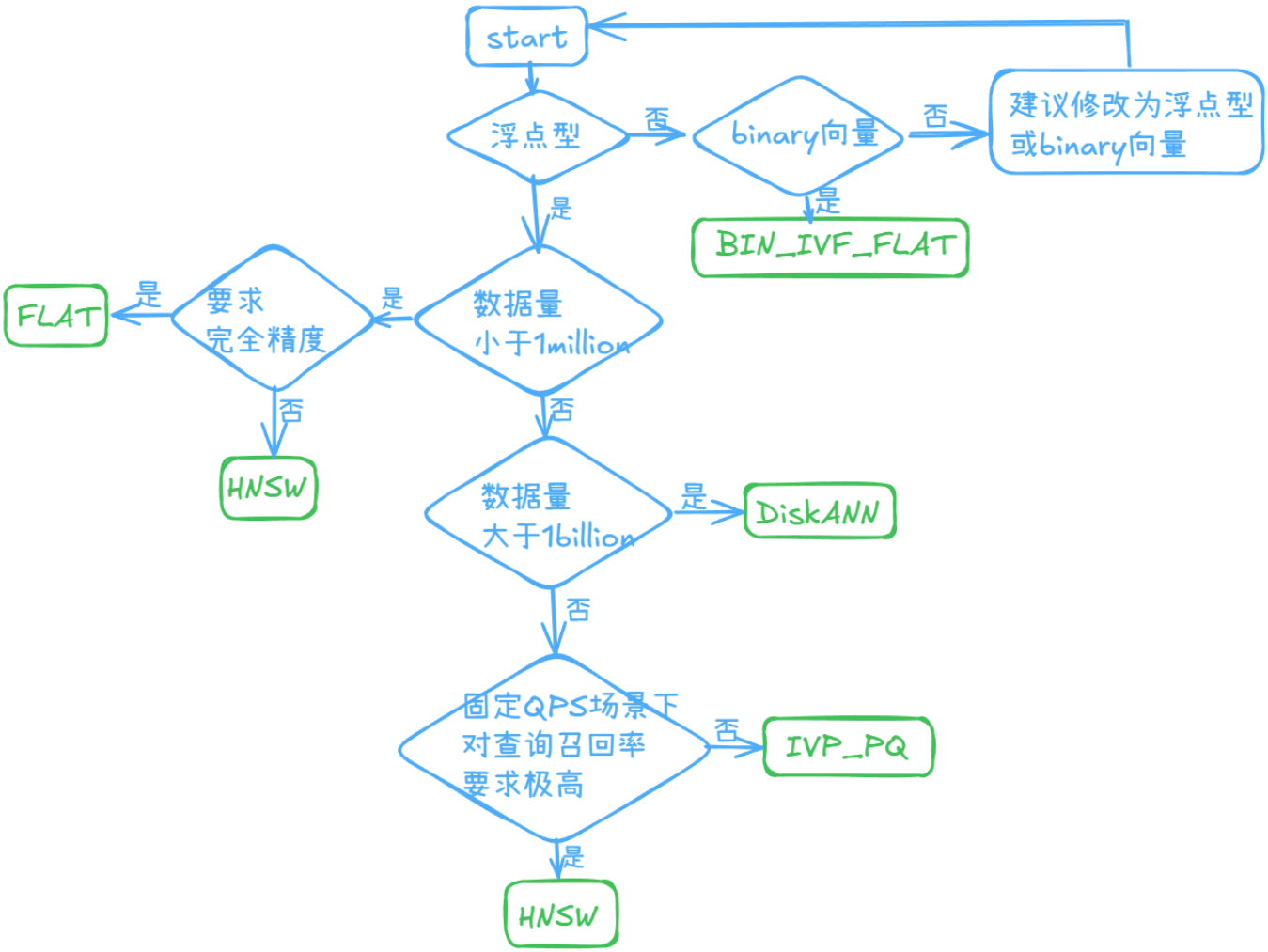
DiskANN的优势：

- 与基于内存的算法相比：如HNSW和IVF，DiskANN在资源消耗和可扩展性上有明显优势，能够在更低的资源消耗下提供相似的查询性能。
- 与基于聚类压缩的算法相比：如IVF_PQ，DiskANN在召回率和性能上保持高效，同时避免了因压缩而导致的召回率降低

## 总结与建议

向量索引技术在大规模、高维度的非结构化数据检索中扮演了至关重要的角色。通过多种创新算法，不同场景中的检索效率得到了显著提升。这些索引技术有效解决了传统方法在处理海量数据时的局限，支持了高效的近似最近邻（ANN）搜索，尤其在LLM、推荐系统、多模态搜索等领域表现出巨大的应用潜力。

然而，选择合适的向量检索方式依赖于具体的应用需求和数据特性，需要在性能和效率之间取得平衡，下图是一些建议：




参考文献：

[1]https://dl.acm.org/doi/pdf/10.5555/3454287.3455520

[2] https://zhuanlan.zhihu.com/p/394393264