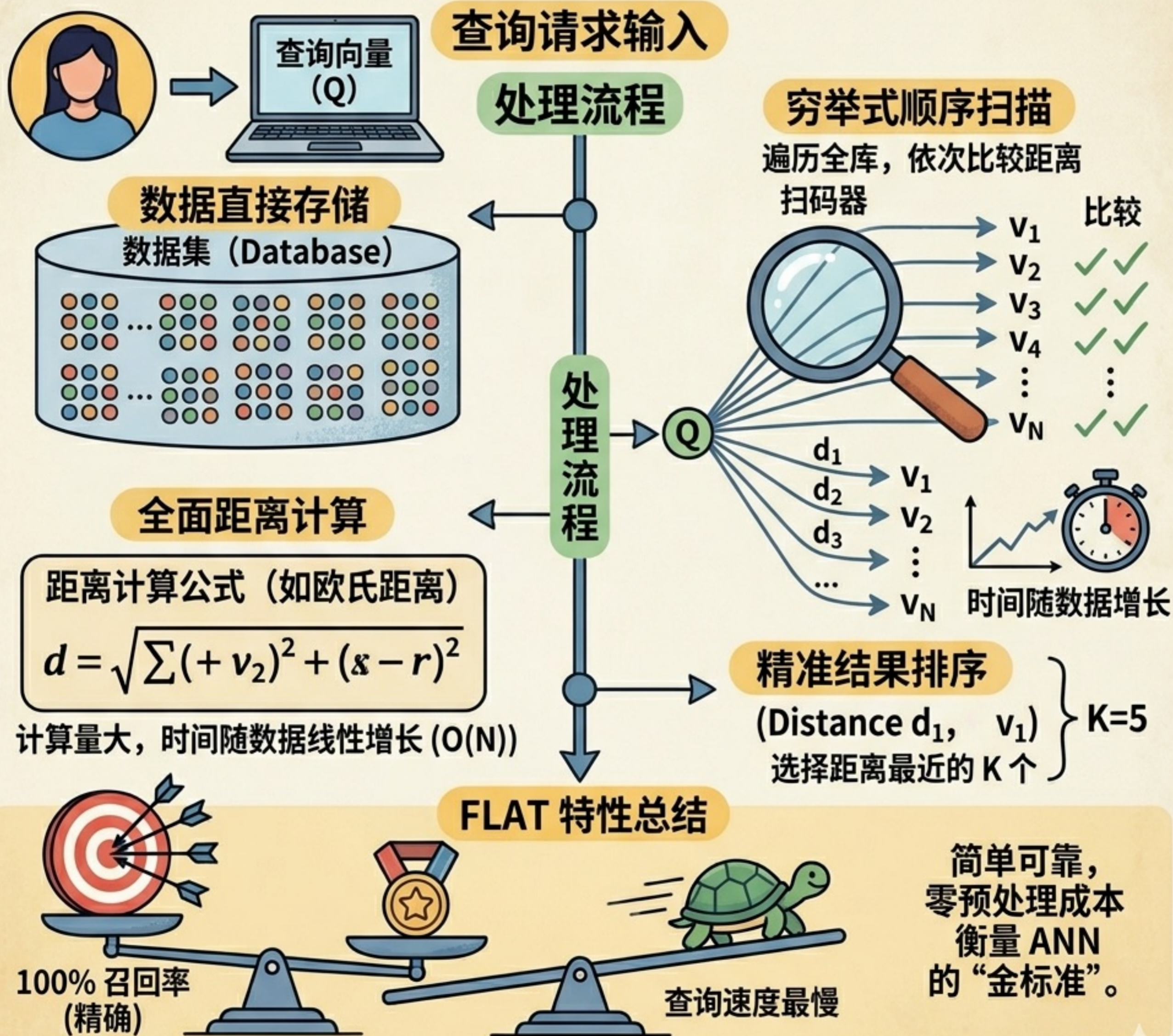


# 向量检索算法：FLAT 蛮力搜索演示流程



FLAT 流程：地毯式搜索，虽慢必达

# LLM (大语言模型) 为什么需要向量数据库?

## 1 过去

关系型数据库 (如 MySQL)



## 2 Embedding 的崛起



## 2 Embedding 的崛起

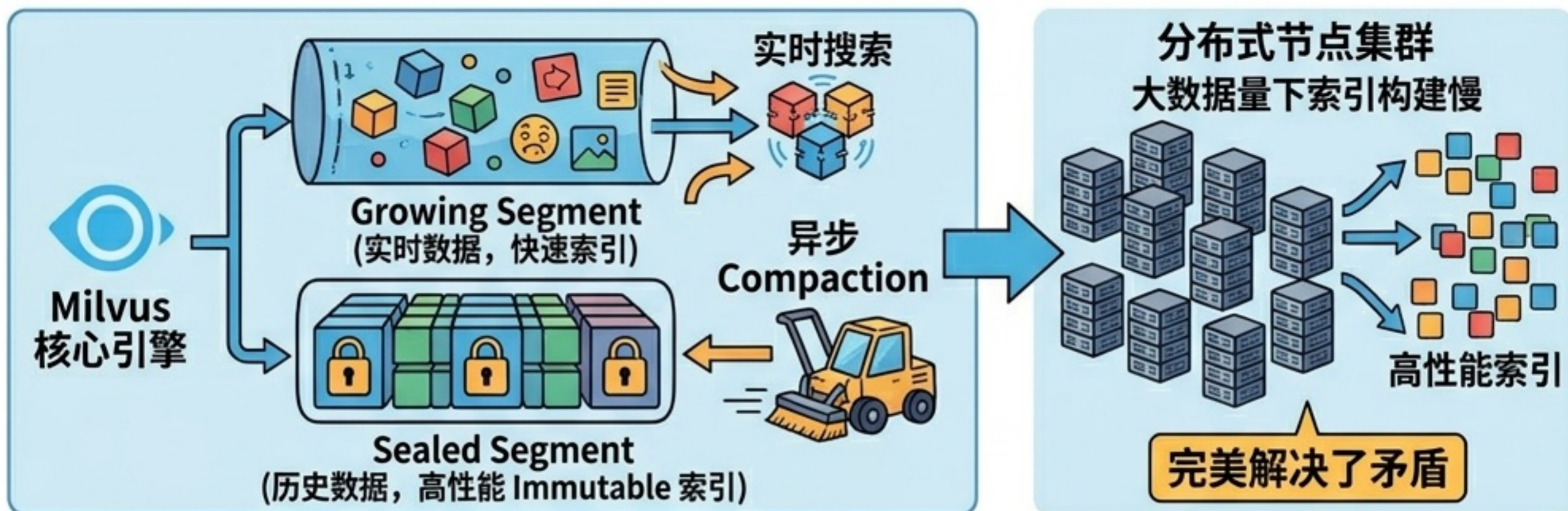


## 3 RAG 场景的刚需



1:1 中文卡通解释图: 向量数据库与 RAG 的强大组合

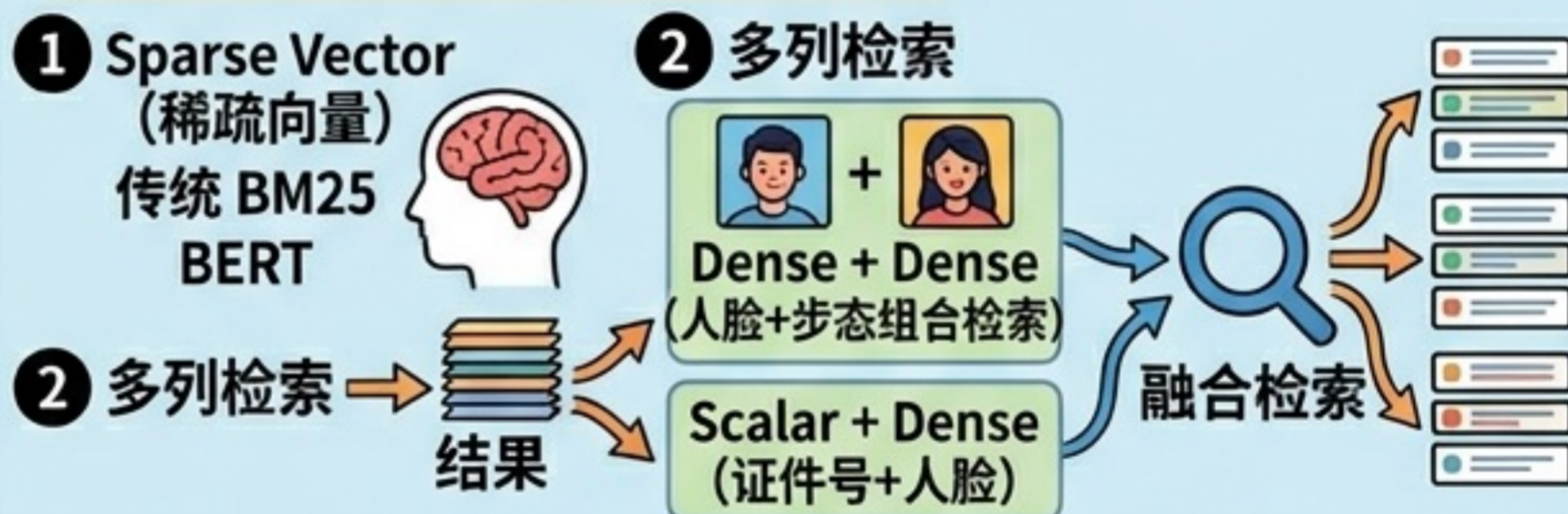
# Milvus 到分布式向量检索



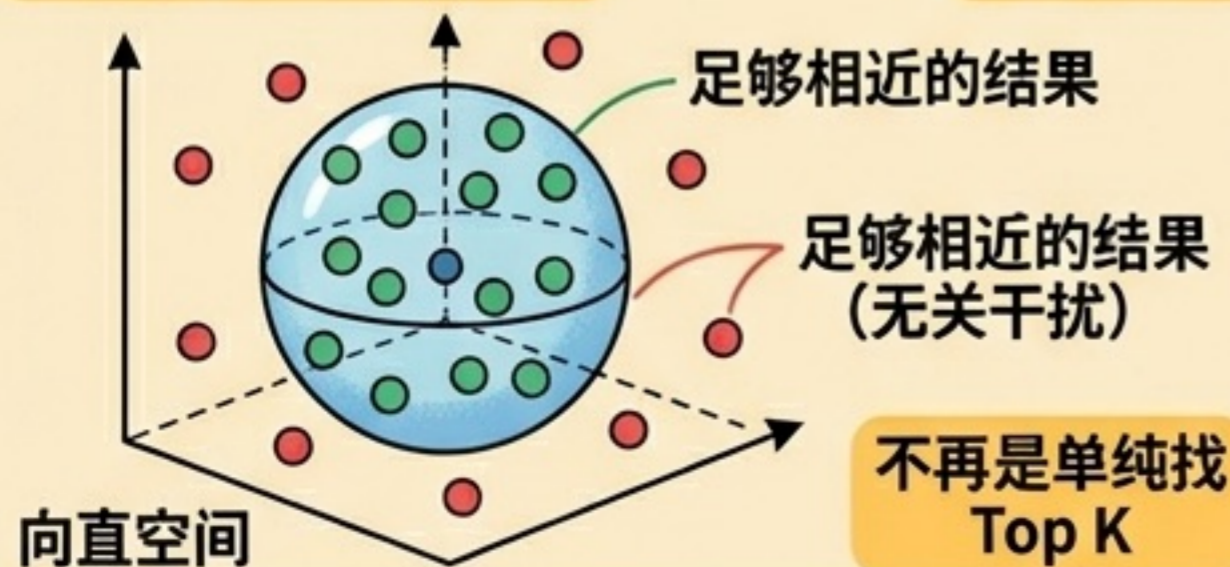
## 传统关键词检索不如



## 混合检索 (Hybrid Search)



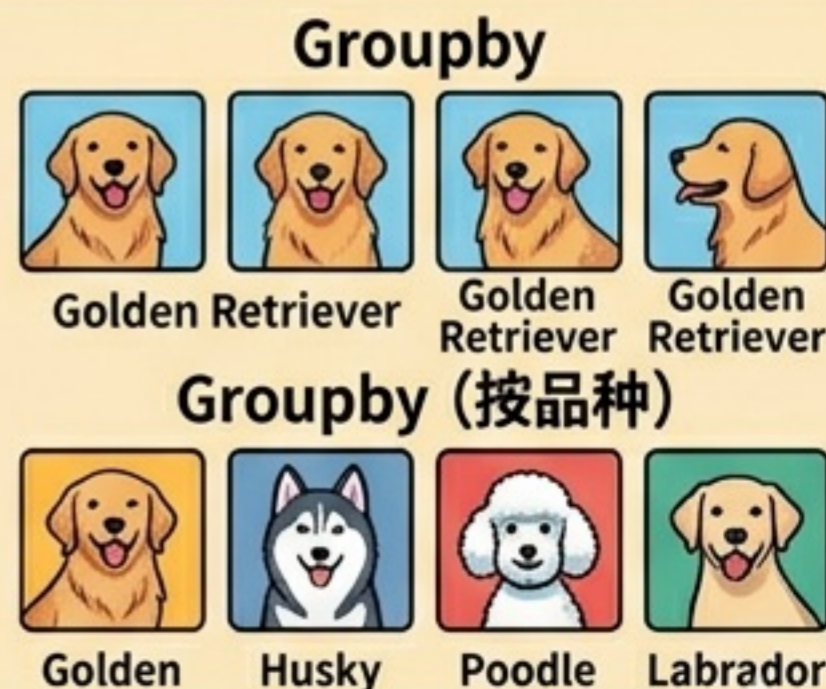
## Range Search



## 丰富的查询语义

搜“狗”

解决结果多样性问题  
解决结果多样防止相似发



## Milvus 技术演进与创新

# 向量数据库 vs 传统数据库：深度融合

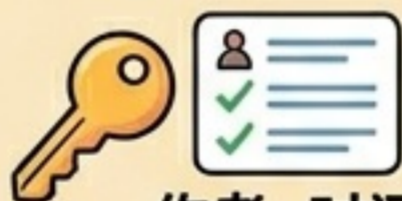
## 传统数据库 (如 MySQL)



典型案例:



MySQL, PostgreSQL



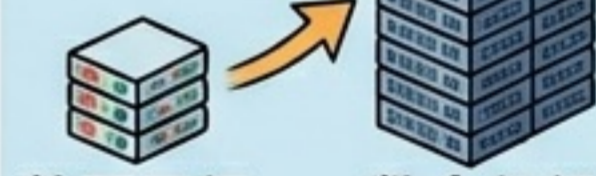
## 向量数据库 (如 Milvus)



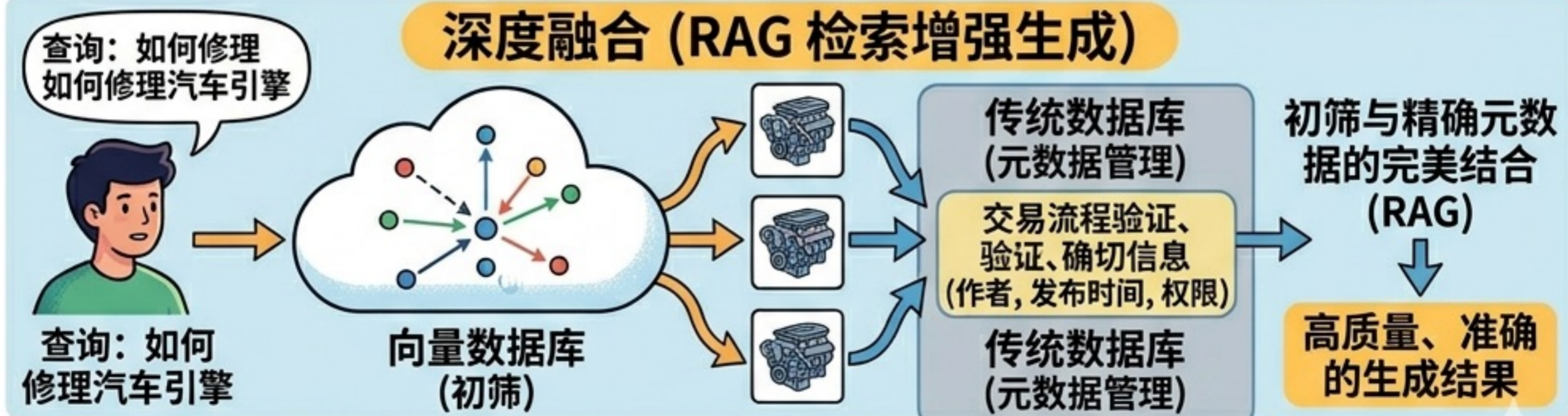
Milvus

案例:  
Milvus, Zilliz, Pinecone

可能性:



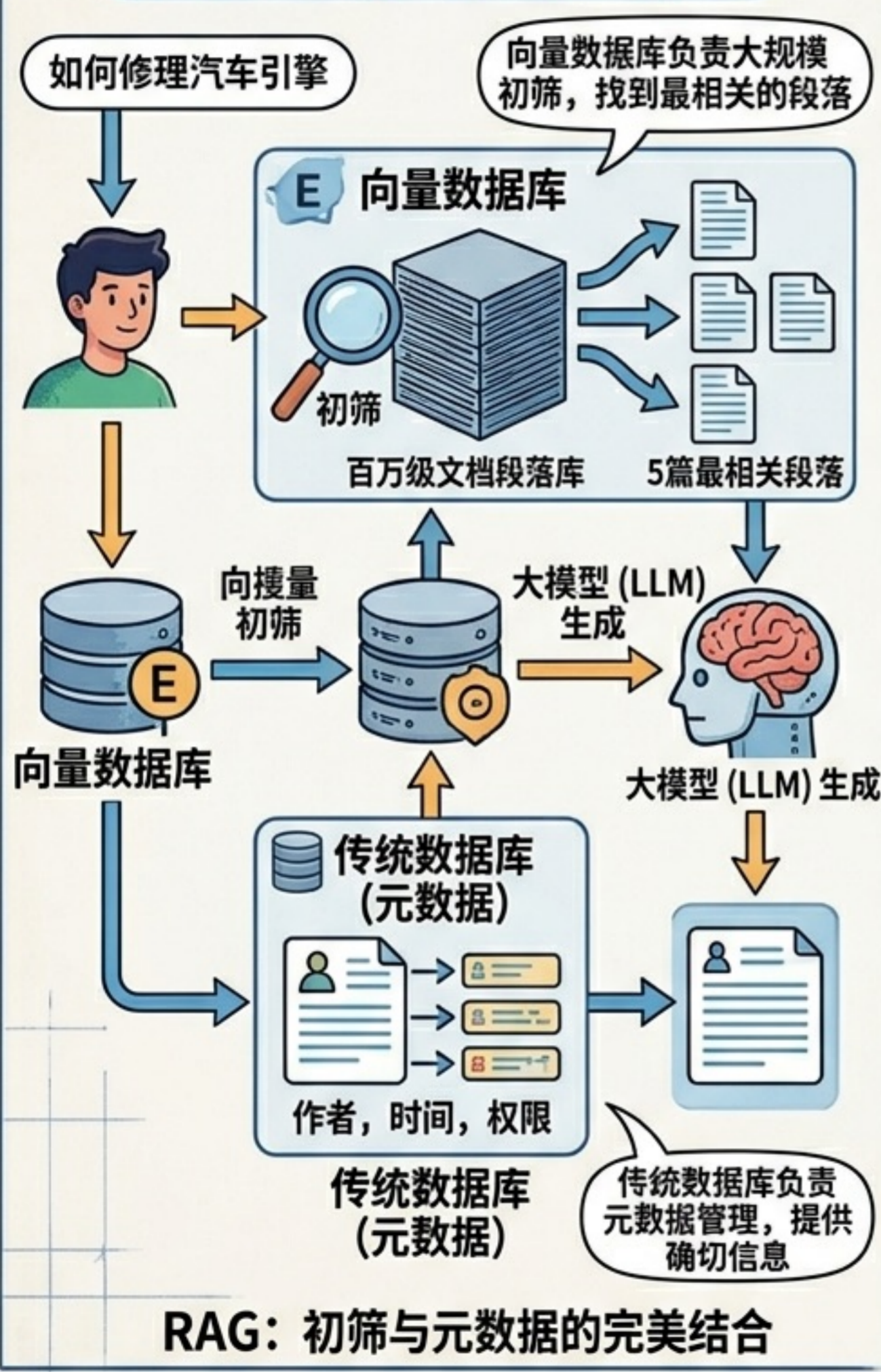
## 深度融合 (RAG 检索增强生成)




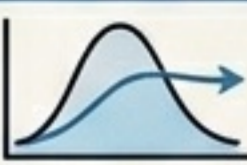

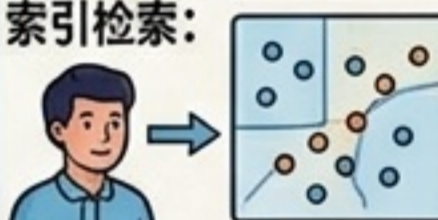



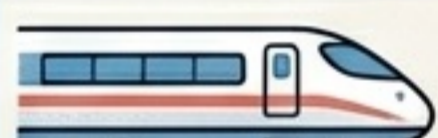



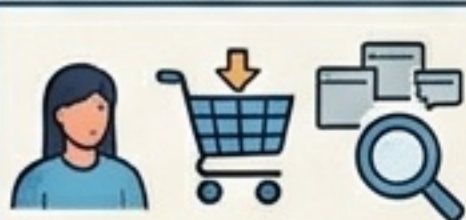
1:1 中文卡通解释图：现代 AI 基础设施的互补组合

# 向量数据库：RAG 的核心与 ANN 检索技术

## RAG 应用中的并存与协同



## KNN 与 ANN 检索技术对比

维度	KNN (K-Nearest Neighbor)	ANN (Approximate Nearest Neighbor)
搜索性质	 精确搜索 (Exarch)	 近似搜索 (Approximate Search)
计算方式	暴力计算:  遍历全库, 计算所有距离	索引检索:  只在候选子集中计算距离
准确率	 100% (理论上的最优解)	 < 100% (存在微小召回损失)
查询速度	 随数据量线性增长 ( $O(N)$ ) 大数据下极慢	 极快 ( $O(\log N)$ 或 $O(1)$ ), 支持亿级数据
内存消耗	 低 (只需存储原始数据)	 额外索引结构 高 (需要额外存储索引结构)
适用场景	 小数据集、对精度要科研	 生产环境、RAG、海量检索

向量数据库：RAG 场景下的深度融合与 ANN 检索的强大性能

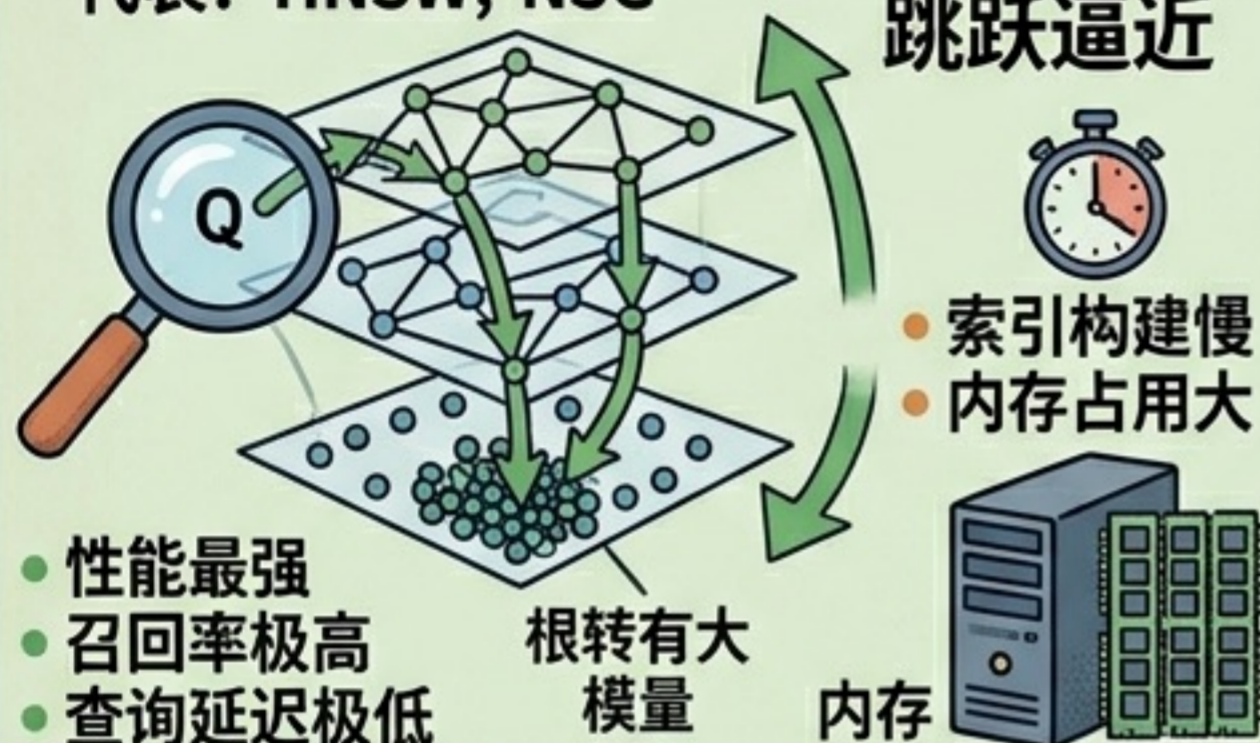
1:1 中文卡通解释图：RAG 场景下的深度融合与深度技术

# 向量检索算法：核心原理与优缺点

索引是附加结构，基于 ANN 算法，虽降低召回（可忽略）但极快。  
本图旨在理解如何最小化成本、最大化效益。

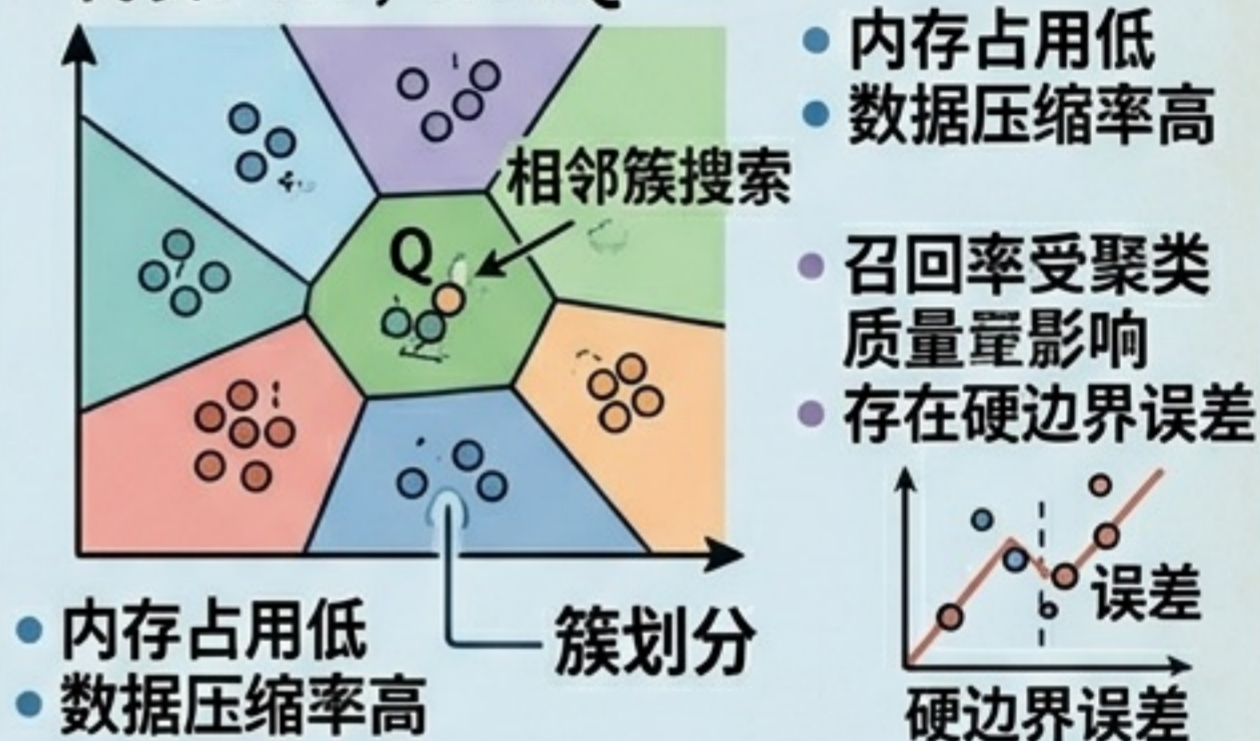
## 基于图 (Graph)

代表：HNSW, NSG



## 基于聚类 (Quantization)

代表：IVF, IVFPQ



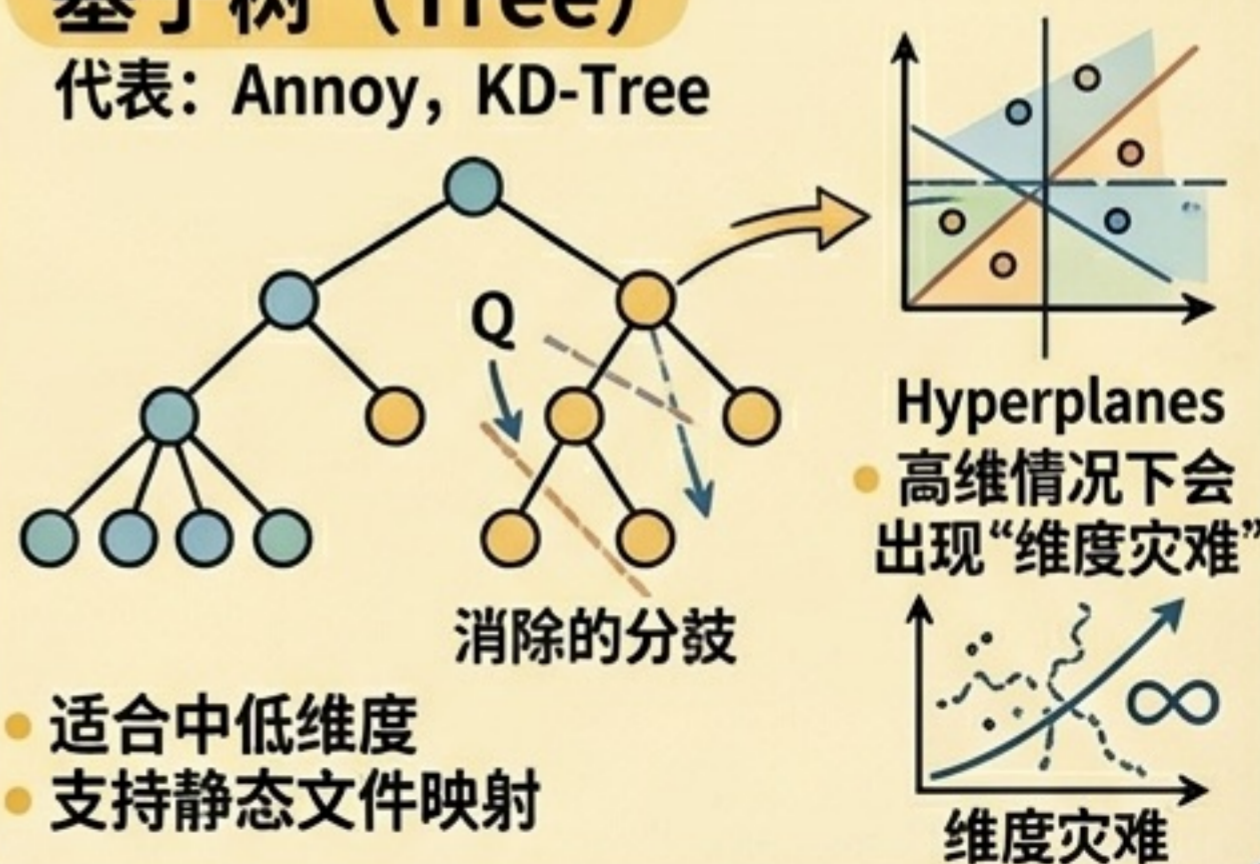
## 基于哈希 (Hash)

代表：LSH



## 基于树 (Tree)

代表：Annoy, KD-Tree



最大限度减少成本，最大限度提高效益：深入理解索引与算法

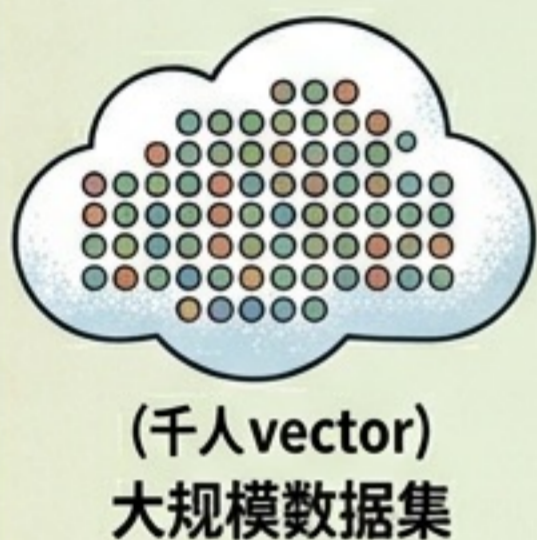
1:1 中文卡通解释图：ANN 算法深度对比与应用

# 向量检索算法：IVF\_FLAT (反转文件扁平) 深度演示流程

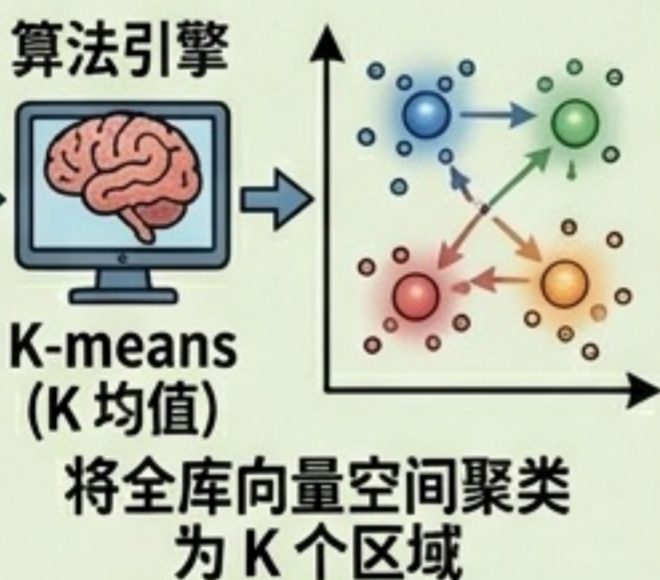
双层方法：通过聚类减少搜索空间，同时保持簇内数据的精确距离计算。  
适用于大规模数据集、高精度度场景。

## 索引构建流程 (聚类与分配)

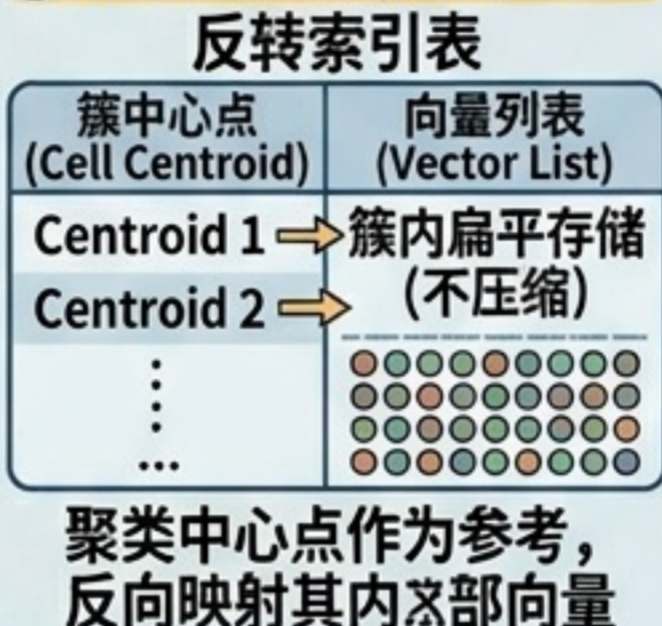
### 1 全库数据



### 2 K-means 聚类



### 3 反转文件 (IVF) 映射

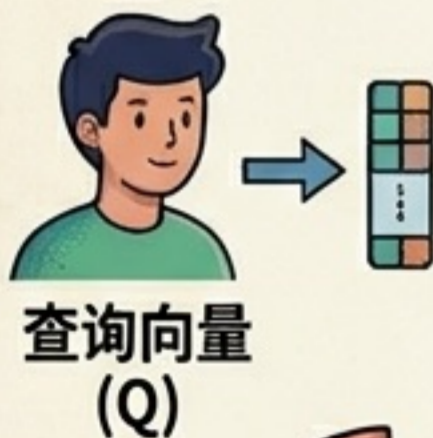


### IVF 就像书籍索引

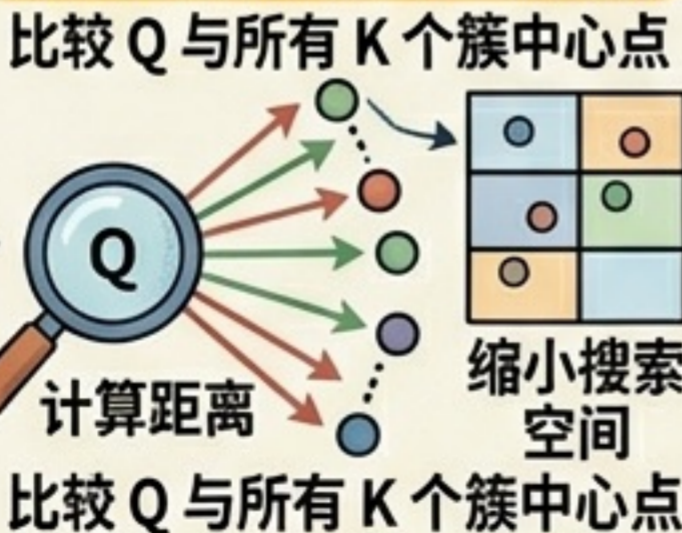


## 查询搜索流程 (二级过滤与排序)

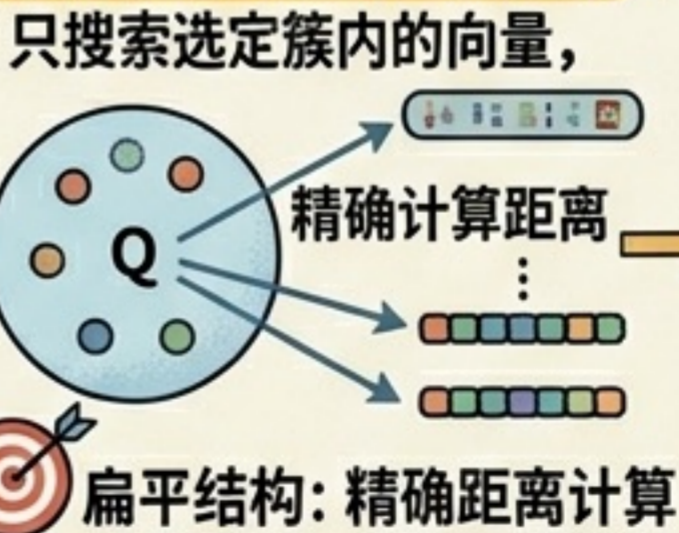
### 4 查询请求



### 5 簇中心点粗筛 (一级搜索)



### 6 候选簇内细筛 (二级搜索)



### 7 结果排序与 Top K



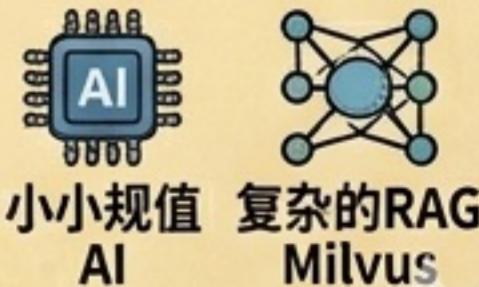
## IVF\_FLAT 特性总结



索引构建成本 (慢)

内存消耗 (低, 无压缩数据, 额外索引占用)

### 适用场景



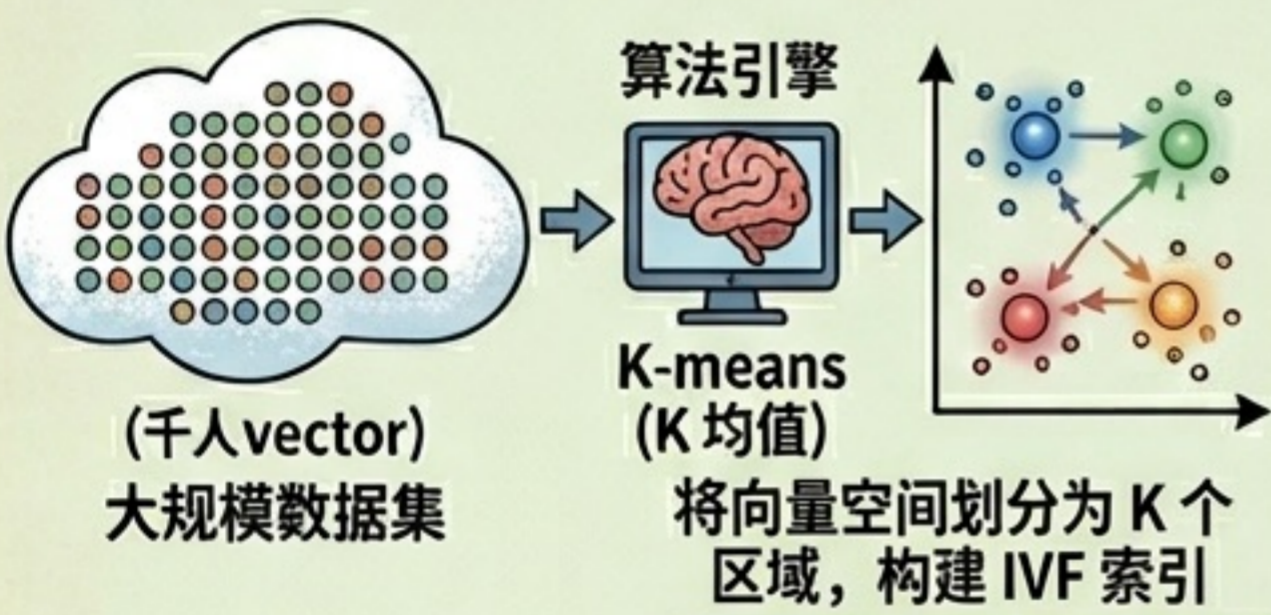
IVF\_FLAT: 聚类加速, 扁平存储, 兼顾速度与精度

# 向量检索算法：IVF\_PQ (反转文件乘积量化) 深度演示流程

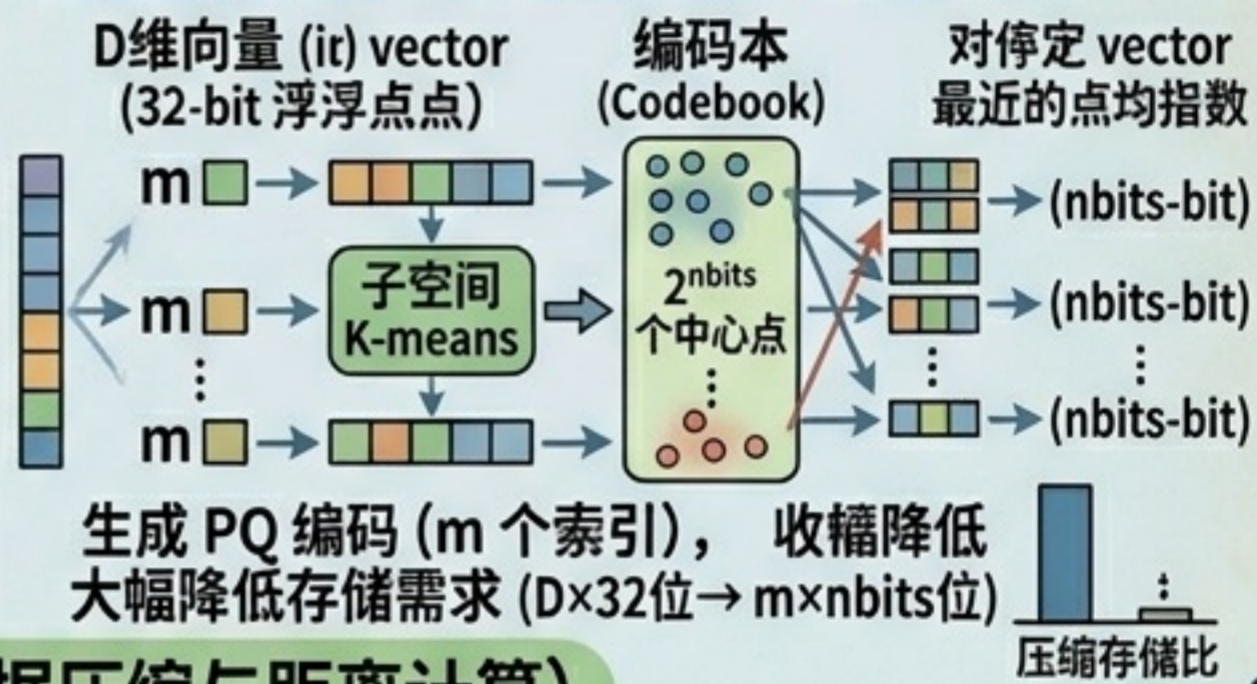
混合方法：IVF 粗筛选缩小范围，PQ 细筛选使用压缩数据快速计算近似距离。适用于内存有限的大型数据集。

## IVF 粗筛选流程 (索引构建与筛选)

### 1 索引构建：K-means 聚类



### 2.1 PQ 数据压缩流程 (索引构建)

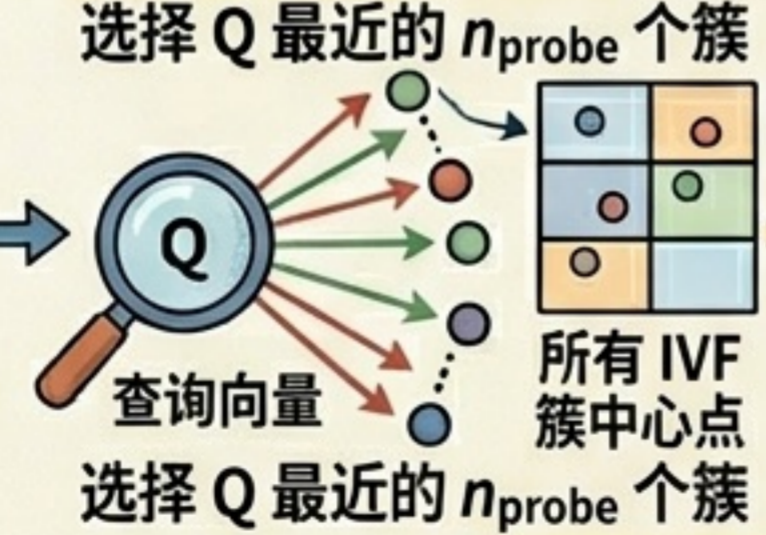


## PQ 细筛选流程 (数据压缩与距离计算)

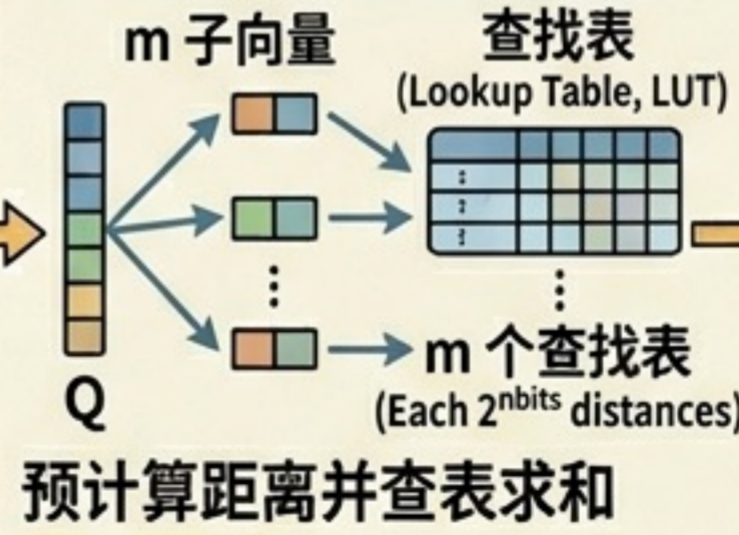
### 2 查询请求



### 3 二级查找 (粗筛)



### PQ 距离计算流程 (查询)



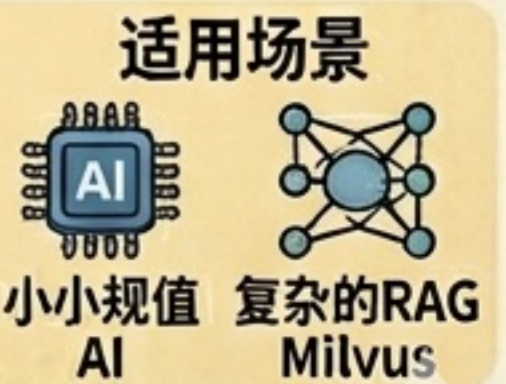
### 4 结果排序与 Top K



## IVF\_PQ 特性总结



- 索引构建成本 (中-慢)
- 内存消耗 (低, 高压压缩数据, 额外索引占比大)



IVF\_PQ: 粗筛选与压缩计算的完美融合，赋能大规模 AI 检索

# 向量检索算法：HNSW（分层导航小世界图）深度演示流程

构建多层图网络，上层远距离跳转，下层细粒度搜索，兼顾高召回与低延迟。

## HNSW 索引结构与层层下降流程

### 查询流程演示

### 多层地图导航原理



第3层  
(Layer 3, 最稀疏)



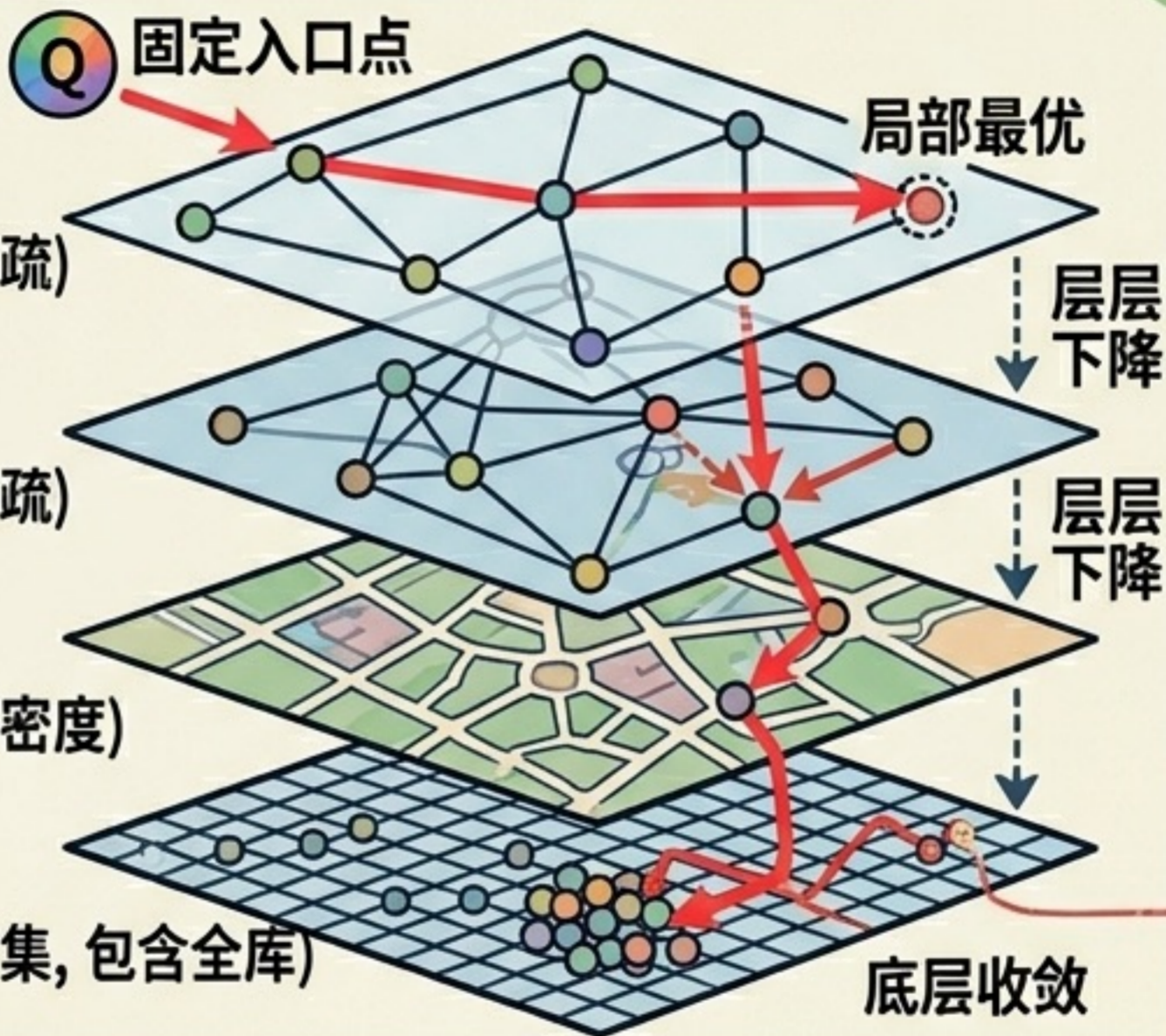
第2层  
(Layer 2, 次稀疏)



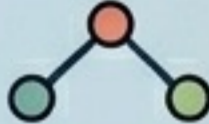
第1层  
(Layer 1, 中等密度)



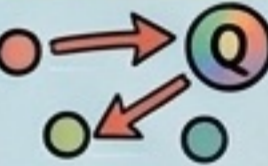
第0层  
(Layer 0, 最密集, 包含全库)



### 入口点



### 贪婪搜索



### 层层下降



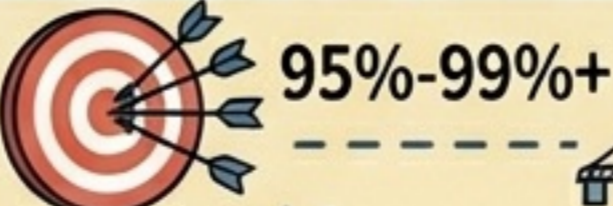
### 最后细化



## HNSW 特性总结

维度	特点
查询速度	极快 ( $O(\log N)$ )
准确率	非常高
内存开销	高
构建时间	较长

维度	特点
查询速度	极快 ( $O(\log N)$ )
准确率	非常高
内存开销	高
构建时间	较长



### 评价



亿级数据下毫秒级延迟

理论召回极高

需存储原始数据  
+ 边信息 (指针)

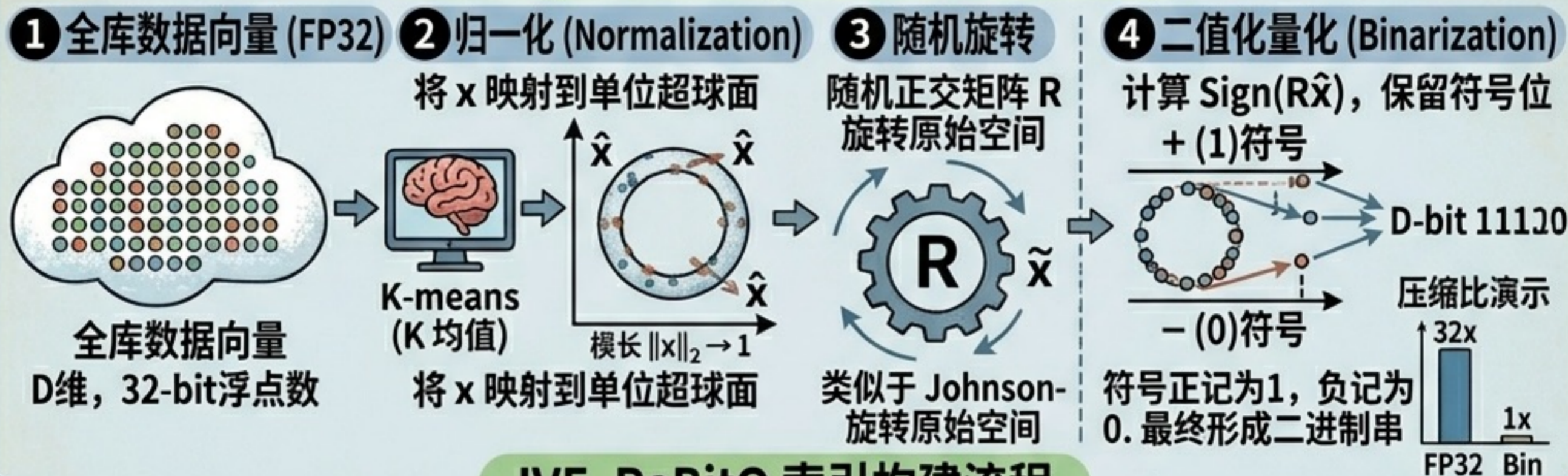
插入新数据需不断  
寻找邻居并建立连接

HNSW 特性总结：分层导图，深度演流程

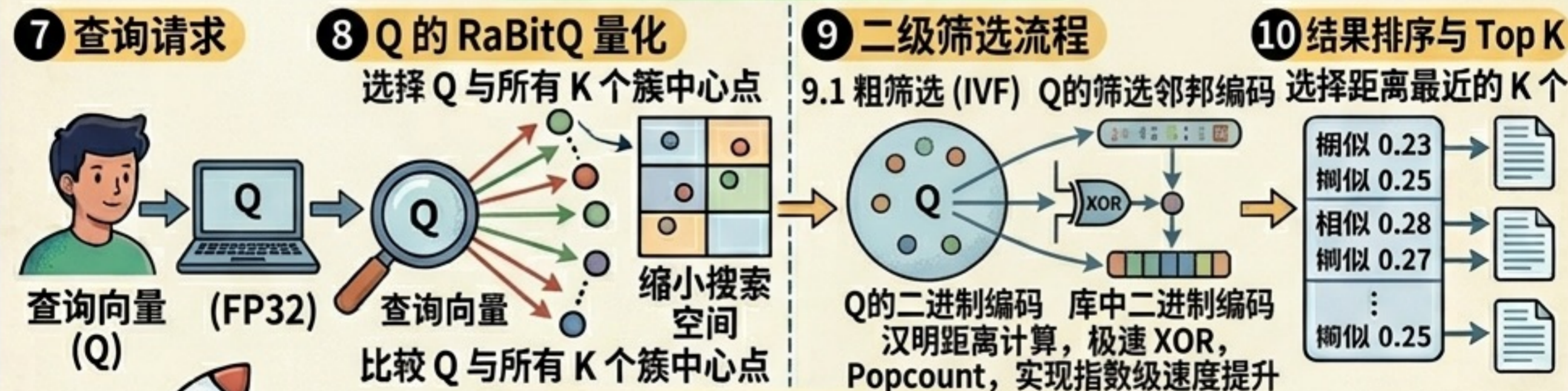
# 向量检索算法：IVF\_RaBitQ (RaBitQ 量化) 深度演示流程

基于几何变换与符号投影的二进制量化，追求极致存储与速度，适用于内存极度受限的大数据集。

## RaBitQ 量化过程 (随机旋转与符号投影)



## IVF\_RaBitQ 索引构建流程



## IVF\_RaBitQ 特性总结与应用



快速搜索



近似精度

极致存储  
(1:32 压缩比)

索引构建成本 (中-慢)

内存消耗 (低, 高压缩数据, 额外索引占用)

适用场景



小小规模 AI



复杂的 RAG  
Milvus

IVF\_RABITQ: 几何压缩与二进制计算的完美融合, 赋能大规模 AI 检索