HilbertClient API 文档

1. 概述

`HilbertClient` 是与 Hilbert 服务端通信的 C++ 客户端封装，支持以下功能：

- 索引管理：创建、删除、列举所有索引
- 数据操作：训练（train）、添加（add）、删除（remove)、查询（query)、更新（update)、搜索（search)

所有方法返回 `int32_t`，`0` 表示成功，非 `0` 表示失败。

2. 数据类型

2.1 枚举 `IndexType`

值	描述
BF	Brute-Force 索引
IVF	Inverted File 索引

```
enum class IndexType {
    BF  = 0,  /**< Brute-Force 索引 */
    IVF = 1   /**< Inverted File 索引 */
};
```

2.2 结构体 `Index`

```
struct Index {
    std::string name;
    uint32_t    nlist;
    uint32_t    dim;
    uint32_t    nb;
    IndexType   index_type;
    uint32_t    replica_num;
};
```

字段	类型	说明
`name`	`std::string`	索引名称，字母/数字/下划线组成，长度 `[1,50]`

`nlist`	`uint32_t`	子索引数（IVF 簇数），范围 `[1,262144]`，且 `nlist*39 <= nb`
`dim`	`uint32_t`	向量维度，范围 `[1,8192]`
`nb`	`uint32_t`	向量数量，nb最大值<6 * 1024^3 / 2 / dim * 0.95 * card_num
`index_type`	`IndexType`	索引类型
`replica_num`	`uint32_t`	副本数，范围 `[0,2]`

2.3 别名

```
using vec_id_t   = uint32_t;  // 向量 ID
using vec_size_t = uint32_t;  // 向量数量
```

3. 类 `HilbertClient`

```
class HilbertClient {
public:
    HilbertClient();
    virtual ~HilbertClient();
    ...
};
```

3.1 方法一览

方法	简要说明
`init`	初始化客户端
`create_index`	创建新索引
`delete_index`	删除索引
`query_all_index`	列举所有索引
`train`	训练索引
`add`	向索引中添加向量
`remove`	从索引中删除单个向量
`query`	根据 ID 查询向量

`update`	更新已有向量
`search`	向量搜索

3.2 详细说明

`init`

```
int32_t init(
    const std::string& server_address,
    const int32_t timeout_ms = -1,
    const int32_t log_level  = 3
);
```

参数	类型	说明
server_address	`std::string`	服务端地址，格式 `"ip:port"` 或 `"domain:port"`
timeout_ms	`int32_t`	请求超时时间（毫秒），`-1` 表示不超时

log_level	`int32_t`	日志级别：
		`0: trace`
		`1: debug`
		`2: info`
		`3: warn`
		`4: err`
		`5: critical`
		`6: off`

返回：`0` 成功；非 `0` 失败

`create_index`

```
int32_t create_index(
    const std::string& name,
    const uint32_t dim,
    const uint32_t replica_num,
    const IndexType index_type,
    const uint32_t card_num
);
```

参数	类型	说明
name	`std::string`	索引名称，字母/数字/下划线组成，长度 `[1,50]`

dim	`uint32_t`	向量维度，范围 `[1,8192]`
replica_num	`uint32_t`	副本数，范围 `[0,2]`
index_type	`IndexType`	索引类型
card_num	`uint32_t`	卡数量，范围 `[1,8]`

前置条件：

- `1 <= name.size() <= 50` 且 仅包含字母、数字、下划线
- `1 <= dim <= 8192`
- `0 <= replica_num <= 2`
- `1 <= card_num <= 8`

返回：`0` 成功；非 `0` 失败

`delete_index`

```
int32_t delete_index(const std::string& name);
```

参数	类型	说明
name	`std::string`	索引名称

返回：`0` 成功；非 `0` 失败

`query_all_index`

```
int32_t query_all_index(std::vector<Index>& indices);
```

参数	类型	说明
indices	`std::vector<Index>&`	输出：所有索引描述列表

返回：`0` 成功；非 `0` 失败

`train`

```
int32_t train(
```

```
    const std::string& name,
    const vec_size_t nb,
    const uint32_t dim,
    const float* xbs,
    const uint32_t nlist,
    const int32_t timeout_ms = -1
);
```

参数	类型	说明
name	`std::string`	索引名称
nb	`vec_size_t`	向量数量，nb最大值 < 6 * 1024^3 / 2 / dim * 0.95 * card_num
dim	`uint32_t`	向量维度
xbs	`const float*`	输入向量数据，长度 = `nb * dim`
nlist	`uint32_t`	IVF 簇数，范围 `[1,262144]`，且 `nlist * 39 >= nb`
timeout_ms	`int32_t`	请求超时时间（毫秒），`-1` 表示不超时

前置条件：

- `nb <= 6 * 1024^3 / 2 / dim * 0.95 * card_num`
- `1 <= nlist <= 262144` 且 `nlist * 39 <= nb`
- `1 <= dim <= 8192`

返回：`0` 成功；非 `0` 失败

---

add

```
int32_t add(
    const std::string& name,
    const vec_size_t nb,
    const uint32_t dim,
    const float* xbs,
    std::vector<vec_id_t>& ids,
    const int32_t timeout_ms = -1
);
```

参数	类型	说明
name	`std::string`	索引名称

nb	`vec_size_t`	向量数量
dim	`uint32_t`	向量维度
xbs	`const float*`	要添加的向量数据，长度 = `nb * dim`
ids	`std::vector<vec_id_t>&`	输出：返回的向量 ID 列表
timeout_ms	`int32_t`	请求超时时间（毫秒），`-1` 表示不超时

前置条件：

- `nb <= 6 * 1024^3 / 2 / dim * 0.95 * card_num`
- `1 <= dim <= 8192`

返回：`0` 成功；非 `0` 失败

`remove`

```
int32_t remove(const std::string& name, const vec_id_t id);
```

参数	类型	说明
name	`std::string`	索引名称
id	`vec_id_t`	向量 ID，**必须**是 `add` 返回

返回：`0` 成功；非 `0` 失败

`query`

```
int32_t query(const std::string& name, const vec_id_t id, std::vector<float>&
data);
```

参数	类型	说明
name	`std::string`	索引名称
id	`vec_id_t`	向量 ID，**必须**是 `add` 返回
data	`std::vector<float>&`	输出：向量数据，长度 = `dim`

返回：`0` 成功；非 `0` 失败

`update`

```
int32_t update(const std::string& name, const vec_id_t id, const uint32_t dim,
const float* data);
```

参数	类型	说明
name	`std::string`	索引名称
id	`vec_id_t`	向量 ID，**必须**是 `add` 返回
dim	`uint32_t`	向量维度
data	`const float*`	新的向量数据，长度 = `dim`

前置条件：

- `1 <= dim <= 8192`

返回：`0` 成功；非 `0` 失败

`search`

```
int32_t search(
    const std::string& name,
    const vec_size_t nq,
    const uint32_t dim,
    const float* query,
    const uint32_t nprobe,
    const uint32_t k,
    std::vector<float>& distances,
    std::vector<vec_id_t>& ids
);
```

参数	类型	说明
name	`std::string`	索引名称
nq	`vec_size_t`	查询数量，范围 `[1,1000]`
dim	`uint32_t`	向量维度
query	`const float*`	查询数据，长度 = `nq * dim`
nprobe	`uint32_t`	探索簇数，不超过 `nlist`

k	`uint32_t`	返回最近邻居数，范围 `[1,4096]`
distances	`std::vector<float>&`	输出：距离列表，长度 = `nq * k`
ids	`std::vector<vec_id_t>&`	输出：ID 列表，长度 = `nq * k`

前置条件：

- `1 <= nq <= 1000`
- `1 <= k <= 4096`
- `nprobe <= nlist`
- `1 <= dim <= 8192`

返回：`0` 成功；非 `0` 失败

> **备注**：
>
> - 所有 `id` 参数（`remove` / `query` / `update`）**必须**来源于 `add` 返回的 `ids`。
> - IVF 索引时，`nlist` 为簇数，`nprobe` 为搜索时探索的簇数。