MEP 15 -- Support ScaNN index

Current state: `Under Discussion`

ISSUE: #2771

PRs: `TODO`

Keywords: Vector Index

Released: `TBD`

Summary

We got requirement for supporting ScaNN index in Milvus. This proposal is about how to integrate a new index into knowhere and expose it to client.

Document about how knowhere works in milvus v1.x: m1_chain.pdf

Docement about how knowhere works in milvus v2.0:

https://github.com/milvus-io/milvus/blob/master/docs/design_docs/index_design.md

https://github.com/milvus-io/milvus/blob/master/docs/design_docs/milvus_create_index_en.md

Motivation

Google recently released a new algorithm to find Approximate Nearest Neighbors called ScaNN. They show through benchmarks that it can perform significantly better than the existing solutions like Annoy, FAISS, hnsw. The goal is to add support for ScaNN algorithm to the list of Milvus supported indexes.

Public Interfaces

Design Details

As part of third party library, we will have to add a new library for ScaNN. This directory here: https://github.com/google-research/google-research/tree/master/scann/scann/scann_ops contains APIs for build/search etc. Also, ScaNN uses Bazel as it's dependency management while we would need to use CMake.

Follwing interfaces would be updated in knowhere =>

IndexType.h

```
namespace IndexEnum {
        extern const char* INDEX_SCANN;
}
```

IndexType.cpp

```
namespace IndexEnum {
        const char* INDEX_SCANN = "SCANN";
}
```

IndexType.cpp

```
namespace IndexParams {
        // SCANN params
        constexpr const char* num_leaves = "num_leaves";
        constexpr const char* num_leaves_to_search = "num_leaves_to_search";
        constexpr const char* dimensions_per_block = "dimensions_per_block";
        constexpr const char* scoring = "scoring";
        constexpr const char* quantize = "quantize";
        constexpr const char* reordering_num_neighbors = "reordering_num_neighbors";
}
```

We will create new wrapper classes called IndexScann.h and IndexScann.cpp in knowhere.

IndexScann.h

```
class IndexScann : public VecIndex {

public:
    IndexScann() {
        index_type_ = IndexEnum::INDEX_SCANN;
    }

    BinarySet
    Serialize(const Config& config) override;

    void
    Load(const BinarySet& index_binary) override;

    void
    Train(const DatasetPtr& dataset_ptr, const Config& config) override;

    void
    AddWithoutIds(const DatasetPtr&, const Config&) override;

    DatasetPtr
    Query(const DatasetPtr& dataset_ptr, const Config& config, const faiss::BitsetView bitset) override;

    int64_t
    Count() override;

    int64_t
    Dim() override;

    void
    UpdateIndexSize() override;

}
```

IndexScann.cpp would be the implementation of IndexScann.h.

VecIndexFactory.cpp

IndexType.cpp

```
if (type == IndexEnum::INDEX_SCANN) {
    return std::make_shared<knowhere::IndexSCANN>();
```

ConfAdapter.cpp/ConfAdapter.go: Add conf adapters for ScaNN params(described above)

Compatibility, Deprecation, and Migration Plan

N/A as it's a net new index type to be released in newer versions.

Test Plan

Rejected Alternatives

References

scann