DBMS Final Project

Team 9

組員： 109020021 蘇暐中、109020005 王昊平、109021134 王韋程

Implementations

To extend a relational database system to support storing and querying over vectors.

Adding Bench

我們首先使用了IVF-FLAT Indexing方法來對向量進行分群與計算等操作。

```
//bench/src/main/java/org/vanilladb/bench/server/param/ann/AnnTe
INDICES_DDL[0] = "CREATE INDEX items_idx ON items(i_emb) USING i

public String[] getIndexSchemas() {
        return INDICES_DDL;
    }
```

```
//bench/src/main/java/org/vanilladb/bench/server/param/sift/Sift
public void prepareParameters(Object... pars) {
    numItems = (Integer) pars[0];
    TABLES_DDL[0] = "CREATE TABLE " + getTableName() + " (i_id I
    INDEXES_DDL[0] = "CREATE INDEX " + getIdxName() + " ON " + g
                + ") USING ivf_flat";
    }
```

我們在`AnnTestbedLoaderProc.java`、`SiftTestbedLoaderProc.java`文件中重新啟用了索引創建過程，確保在初始化模式時正確創建新的IVF-FLAT索引。

```
private void createSchemas() {
    // 跳過添加索引
```

```
    if (logger.isLoggable(Level.INFO)) {
        logger.info("Creating indexes...");
    }

    // 創建索引
    for (String sql : paramHelper.getIndexSchemas()) {
        StoredProcedureUtils.executeUpdate(sql, tx);
    }

    if (logger.isLoggable(Level.FINE)) {
        logger.info("Finish creating schemas.");
    }
}
```

Adding Parse

新增ivf_flat的keyword到`Lexer.java`與`Parser.java`中，以及index type的判定到`Index.java`內。

Adding Ivf_flat

在`Cluster.java`中管理與建立centroid/cluster，並在`VectorPair.java`中實作向量數據的管理。

每一個cluster會有一個centroid，目前尚未實現真的centroid，是隨機取一個當作centroid，之後的insertion會比較和每一個cluster的centroid最近者，並插入其中。

```
//Cluster.java
// Create new cluster when inserting
 public class Cluster {

    private int cid;
    private VectorPair centroid;
    private List<VectorPair> vecs;
    private static final String SCHEMA_RID_ID = "id";
    private static final String SCHEMA_RID_BLK = "blk";
    private static final String SCHEMA_RID_VEC = "i_emb";
```

```
    private TableInfo ti;


    // Create new cluster when inserting
    public Cluster(RecordId rid, VectorConstant centroid, int ci
        this.cid = cid;
        this.centroid = new VectorPair(centroid, rid.id(), rid.l
        this.vecs = new ArrayList<>();
        this.vecs.add(this.centroid);

        String newClusterTable = "cluster_" + cid;
        VanillaDb.catalogMgr().createTable(newClusterTable, sche
        ti = VanillaDb.catalogMgr().getTableInfo(newClusterTable
        RecordFile rf = ti.open(tx, false);
        rf.insert();
        rf.setVal(SCHEMA_RID_ID, new IntegerConstant(rid.id()));
        rf.setVal(SCHEMA_RID_BLK, new BigIntConstant(rid.block()
        rf.setVal(SCHEMA_RID_VEC, centroid);
        rf.close();
    }
    ...
    }
```

```
//VectorPair.java
public class VectorPair {
    public VectorConstant VectorConst;
    public Integer rid;
    public Long blk;

    public VectorPair(VectorConstant VectorConst, int rid,long l
        this.VectorConst = VectorConst;
        this.rid = rid;
        this.blk = blk;
```

```
    }
    ...
```

在`IVFFlatIndex.java`實作Ivf_flat indexing，包含創建Indexing、Insert、search、delete、close等。

```
//創建ivf flat indexing
public class IVFFlatIndex extends Index {
...
        public IVFFlatIndex(IndexInfo ii, SearchKeyType keyType,
                super(ii, keyType, tx);
                this.ccMgr = tx.concurrencyMgr();
                this.ii = ii;
                // System.out.println("Init");

                // this.ti = getIndexUsedTableInfo(ii.indexName(
                rwLock.writeLock().lock();
                if (isBench && listCluster.isEmpty()) {
                    System.out.println("Load cluster");
                    for (int i = 0; i < NUM_CLUSTERS_MAX; i++) {
                        Cluster cluster = new Cluster(i, tx);
                        listCluster.add(cluster);
                    }
                    System.out.println("Load cluster complete");
                }
                rwLock.writeLock().unlock();

            }
    ...
    }
```

新增`ivfFlatIndexPlan.java`和`ivfFlatIndexScan.java`來針對ivfFlatIndex創建和管理搜尋計畫與進行Index搜尋結果。

```
public class IvfFlatIndexPlan implements Plan {
    private TablePlan tp;
```

```
    private IndexInfo ii;
    private DistanceFn distFn;
    private Transaction tx;
    private Histogram hist;

    public IvfFlatIndexPlan(TablePlan tp, IndexInfo ii, Distance
        this.tp = tp;
        this.ii = ii;
        this.distFn = distFn;
        this.tx = tx;
        HashMap<String, ConstantRange> searchRange = new HashMap
        searchRange.put("i_emb", new VectorConstantRange(distFn

        hist = SelectPlan.constantRangeHistogram(tp.histogram(),
    }
...
}
```

```
public class IvfFlatIndexScan implements UpdateScan {
    private IVFFlatIndex idx;
    private TableScan ts;
    private DistanceFn distFn;

    public static final int NUM_CLUSTERS_MAX = CoreProperties.ge
            IVFFlatIndex.class.getName() + ".NUM_CLUSTERS_MAX",

    public IvfFlatIndexScan(Index idx, DistanceFn distFn, TableS
        this.idx = (IVFFlatIndex) idx;
        this.distFn = distFn;
        this.ts = ts;
    }
...
}
```

同時也在`TablePlanner.java`確保能呼叫到Ivf_flat indexing。

```
...
public Plan makeSelectPlan() {
    Plan p = makeIndexSelectPlan();
    if (p == null)
        p = tp;
    p = addSelectPredicate(p);
    if (embField == null)
        return p;

    List<IndexInfo> iis = VanillaDb.catalogMgr().getIndexInfo(tl
    IndexInfo ii = null;
    for (IndexInfo iii : iis) {
        if (iii.fieldNames().contains(embField.fieldName()))
            ii = iii;
    }
    if (ii == null)
        throw new RuntimeException("Can't find index for vector
    p = new IvfFlatIndexPlan(tp, ii, embField, tx);
    p = new NearestNeighborPlan(p, embField, tx);
    return p;
}
...
```

Future work

本次僅實現較簡單的KNN算法，雖然運行快速，但並未真的實現近似KNN，所以recall表現不佳。未來在改進成效時，可在centroid的設定上多著墨。

SIMD implementation

We haven’t implemented this part.

Experiments

Environment

Intel Core i7-12700 CPU
48 GB RAM
1TB SSD
Debian GNU/Linux 12

Experiment

of txns (including aborted) during benchmark period: 0

ANN - committed: 0, aborted: 0, avg latency: 0 ms
INSERT - committed: 0, aborted: 0, avg latency: 0 ms
Recall: NaN%
TOTAL - committed: 0, aborted: 0, avg latency: 0 ms

original:

of txns (including aborted) during benchmark period: 0

ANN - committed: 0, aborted: 0, avg latency: 0 ms
INSERT - committed: 0, aborted: 0, avg latency: 0 ms
Recall: NaN%
TOTAL - committed: 0, aborted: 0, avg latency: 0 ms

ANN - committed: 0, aborted: 0, avg latency: 0 ms
INSERT - committed: 0, aborted: 0, avg latency: 0 ms
Recall: NaN%
TOTAL - committed: 0, aborted: 0, avg latency: 0 ms

#of txns (including aborted) during benchmark period: 0

ANN - committed: 0, aborted: 0, avg latency: 0 ms
INSERT - committed: 0, aborted: 0, avg latency: 0 ms

Recall: NaN%
TOTAL - committed: 0, aborted: 0, avg latency: 0 ms

IVF-Flat:

#of txns (including aborted) during benchmark period: 3196

ANN - committed: 2851, aborted: 0, avg latency: 201 ms
INSERT - committed: 345, aborted: 0, avg latency: 70 ms
Recall: 7.33%
TOTAL - committed: 3196, aborted: 0, avg latency: 188 ms