Programming Assignment 1: Pseudocode CS5280

Darpan Gaur
CO21BTECH11004

BOCC

variables

```
item -> vector
    stores the value of data items

local_items -> vector
    stores the value of data items local for transaction

read_set -> map(trans_id , vector)
    stores the set of data items read by each transaction
write_set -> map(trans_id , vectore)
    stores the set of data items written by each transaction

commit_set -> vector
    stores the set of transactions that have committed

read_list -> map(data_item , vector)
    stores the list of transactions that have read each data item
write_list -> map(data_item , vector)
    stores the list of transactions that have written each data item

is_aborted -> vector
    true if transaction is aborted , false otherwise
```

begin_trans

```
begin_trans()
{
    // returns the id for the transaction
    lock(id_lock);
    trans_id = id + 1;
    id++;
    initialize: read_set[trans\_id]
    initialize: write_set[trans\_id]
    set is_aborted[trans\_id] = false
    unlock(id_lock);
    return trans_id;
}
```

read(i, x, l)

```
read(i, x, l)
{
    // i is the transaction id
    // x is the variable to be read
    // store value of x in l
    lock(item_lock);
    if (is_aborted[i]==true){
        free_trans(i);
        unlock(item_lock);
        return -1;
    }
    l -> local_items[x]
    read_set[i].push(x);
    read_list[x].insert(i);
    unlock(item_lock);
    return 0;
}
```

write(i, x, l)

```
write(i, x, l)
{
    // i is the transaction id
    // x is the variable to be written
    // l is the value to be written
    lock(item_lock);
    if (is_aborted[i]==true){
        free_trans(i);
        unlock(item_lock);
        return -1;
    }
    update local varible l
    local_items[i] -> l
    write_set[i].push(x);
    write_list[x].insert(i);
    unlock(item_lock);
    return 0;
}
```

try_commit(i)

```
try_commit(i)
{
    // i is the transaction id
    lock(item_lock);
    if (is_aborted[i]==true) {
        free_trans(i);
        unlock(item_lock);
        return a;
    }
    for d_id in read_set[i] {
        for t_id in write_list[d_id] {
            if (commit_set contains t_id) {
                is_aborted[i] = true;
                free_trans(i);
                unlock(item_lock);
                return a;
            }
        }
    }
    for d_id in write_set[i] {
        item[d_id] = l
    }

    update items vector from local_items vector
    that are in write_set[i]
    commit_set.insert(i);
    free_trans(i);
    unlock(item_lock);
    return c;

}
```

free_trans(i)

```
free_trans(i) {
    delete local_items
    delete read_set
    delete write set
    remove i from read_list
    remove i from write_list
}
```