# Minibase Query Processor


## Group:

1) Ebtehal El_Aaraby (2).
2) Remon Hanna Wadie Youssef (21).
3) Shaimaa Mousa (26).
4) Tarek Khaled Ragab (30)

# Part One

**In this part we implement the following operations: Selection, Projection and Simple Join.**
**To do that we needed first to convert the records of data stored in the database into tuples which can be dealt with.**
**So for that we made 3 different scanning which are FileScan, KeyScan and IndexScan.**
**All three inherit from the iterator class which enable us to get all tuples by calling get Next Tuple as following:**

> **For FileScan: we made HeapScan that returns Tuples instead of byte[]'s.**
> **For KeyScan: we made HashScan that returns Tuples instead of RIDs, where HashScan scans the hash index for records having a given search key.**
> **For IndexScan: we made BucketScan that returns Tuples instead of RIDs, where BucketScan scans the whole hash index.**

**After that we made the Selection, Projection and Simple Join classes which simply do their operations as following:**

> **For Selection: it specifies which tuples to retain under a predicate or predicates that connected with AND operation.**
> **For Projection: it extracts columns from a relation and it doesn't eliminate duplicate tuples.**
> **For Join: it implements the SimpleJoin class to provide the Nested Loop Join.**

# Part Two

**In this par we made the Query Evaluation and Optimization since the parser and query checker have already been done, it was quite simple to continue to make the basic operations like CREATE, DROP, INSERT and SELECT using of course part one with these details:**

**For Create: we made Create Index which make an index on some tables exists in the database with the help of QueryCheck class.**

**For Drop: we made a simple class that can drop the indices on the tables.**

**For Insert: we made a class that insert succeefuly in the database with the help of the QueryCheck too.**

**For Select: it was the most important class to implement as we first had to make a good plan for it to cost less time for retrieving data from the database and our plan includes the following:**

- **After checking that every table, column exists in our database we make simple iterators on each table.**
- **Then before making the join we check if some of the predicates belongs to specific table or not if so we modify its iterator to be just for these predicates**

- **After finishing that we simply make the simple join between these iterators based on the conditions provided in predicates.**
- **Finally we make a projection to the required columns or retrieve all columns if * was the required operation.**

**That's how we implement the pushing select by doing the selections first then the Join so as not to cost much time.**