3.

a. Dbxic implements the lazy query processing pipeline.

**- - - - - - - - a - - - - - - - -**

Lazy

parser package parse the queries we write to relational algebra operators in algebra package, then through the optimizer (PlanBuilder) this operators are optimized and transformed to Physical Operators and finally executed. We can deduce PhysicalOperator build the executed code.

PhysicalOperator

*// It masks through a buffer the possible internal multiple-tuple processing to be able to serve tuples one-by-one.*

*/\*\**

*\* PhysicalOperator: Abstract class with the basic operator toolset.*

*\* Given a set of input operators, it builds up an output relation.*

*\* It masks through a buffer the possible internal multiple-tuple*

*\* processing to be able to serve tuples one-by-one.*

*\*/*

**- - - - - - - - b - - - - - - - -**

Yes, at the PlanBuilder.

In the algebra package we have all the operators (Relational Algebra Operatos). When we use the PlanBuilder, we transform the algebra into physicalOperatos. This is the step where we transform that tree that is only composed of algebra operator into an Evaluation Plan.

In our case the optimizer is the class called PlanBuilder, and that optimizer transforms the expression build as things that we have in the package of algebra into things that are in the physicalOperators package, and this is what's actually executed.

*\* It works as follows:*

*\* 1) Decompose the operators into scans, projections, selections and*

*\* joins.*

*\* 2) Drop all unnecessary fields to reduce tuple width, then impose*

*\* selections, enumerate joins, and impose any projection lists and*

*\* sort orders.*

*\*/*

*// Drop all unnecessary fields to reduce tuple width*

*// 2.5) order the joins and cartesian products in a tree*

*// perform sanity check and impose the final projections*

**- - - - - - - - c - - - - - - - -**

c. Least recently used (LRU). We observed that at getBlock() function at BufferManager.java.

BufferManager DBMS

StorageManager is the one that uses the Buffer in order to communicate with the disk, is the bridge between the system itself and the BufferManager.

X Toss-immediate strategy: removes a block as soon as the final tuple of that block is processed

X Most recently used (MRU) strategy: replace the MRU block

Least recently used (LRU) strategy: replace the LRU block ->

private LinkedList<Pair> lReplacement; *// Replacement queue, to know which block to remove if necessary*

*//* *getBlock: returns a block given its identifiers, moving it to the back of the replacement queue.*

*// if it exists, we mark it as recently used by moving it to the back of the replacement queue*

lReplacement.remove(blockid);

lReplacement.add(blockid);

*// move it to the back of the replacement queue*

*// indexToEvict: returns the block to be evicted from the buffer to make room.*

Pair blockid = lReplacement.removeFirst();

**- - - - - - - - d - - - - - - - -**

create table student (id integer, name string, birth\_year integer);

TupleInsertion - public Pipe execute(DBMS engine)

StorageManager -

void castAttributes(Tuple tuple, Relation relation) ... {

TypeCasting.*castValuesToTypes*(tuple.getValues(), relation.getTypes());

}