List of Changes

Lab1

Simpledb/parse/Parser.java	Modify term() routine to call eatOp() in Lexer.java Modify term() to return Term with additional input that is generated from eatOp() i.e. cmp (operator)
Simpledb/parse/Lexer.java	Add an eatOp() to eat the operators ‘!’, ‘>’, ‘<’, ‘=’ and return the operator as string
Simpledb/parse/PredParser.java	Replace lex.eatDelim(‘=’) with lex.eatOp() in term() method
Simpledb/query/Term.java	Modify isStatisfied() routine to evaluate the LHS and RHS with the operator supplied.
Simpledb/query/Expression.java	Modify evaluate() to return the LHS value as the field value. Add an evaluateRHS() to return the value that is being used to compare against LHS.
Simpledb/query/Constant.java	Add greater() comparator to check if LHS > RHS for integers Add lesser() comparator to check if LHS < RHS for integers String is compared using length of string.

Lab2

Simpledb/query/Expression.java	Restore to original implementation.
Simpledb/test/network/CreateStudentDB.java	Create index majorid for student and index studentid for enroll tables after creation of tables and before inserting of data. e.g. “create index majorid on STUDENT(MajorId)”
Simpledb/server/SimpleDB.java	Commented original qp and up. Uncommented heuristic query planner and index update planner
Simpledb/query/Term.java	Removed previous comparison methods and use compareTo for non-equality checks
Simpledb/metadata/IndexInfo.java	Modify open() and blocksAccessed() to use either hash / btree (default) depending on private variable indexmethod which is passed in during IndexInfo initialization.
Simpledb/parse/Parser.java	Modify createIndex() to parse “using hash” or “using btree”. If nothing is specified, btree is used.

Simpledb/parse/CreateIndexData.java	Add a variable call idxmethod and a function call indexMethod() to get idxmethod.
Simpledb/planner/IndexUpdatePlanner.java	Pass indexmethod from CreateIndexData to MetadataMgr
Simpledb/metadata/MetadataMgr.java	Add in an extra parameter indexmethod to createIndex() function. Pass indexmethod to idxmgr.createIndex()
Simpledb/metadata/IndexMgr.java	Add string field 'indexmethod' to schema in IndexMgr() Set string "indexmethod" with indexmethod when createIndex() called. getIndexInfo decides to use hash/btree during query by reading "indexmethod" from schema, then the result is supplied into IndexInfo and store as private variable.

Lab3

Simpledb/opt/HeuristicQueryPlanner.java	Modify step 4 to assign new ProjectPlan to Plan p. Create new SortPlan and assign it to p and return p. Second parameter of SortPlan is data.orders() instead of data.fields() which I created below.
Simpledb/parse/Lexer.java	Add "order" and "by" to keywords.
Simpledb/parse/Parser.java	Create orderList() which parse the strings after the keyword 'order' and 'by'. Works similar to selectList(). Returns a List of <String, String> pairs. Key is field name and value is either asc or desc. Default to asc by wrapping logic in try catch and assigning asc when in catch block. In query(), After getting predicates, I put in logic to extract the keys and values for the orders and pass it to QueryData(fields, tables, pred, orders)
Simpledb/parse/QueryData.java	Add a private List<AbstractMap.SimpleEntry<String, String>> orders. Assign the orders to private orders in constructor. Create a method call orders() which returns orders.
Simpledb/materialize/SortPlan.java	Modify type of sortFields to List<AbstractMap.SimpleEntry<String,String>>.

Simpledb/materialize/RecordComparator.java	Modify type of fields to List<AbstractMap.SimpleEntry<String, String>> In compare() method, sX.getVal(fldname) is replaced with sX.getVal(entry.getKey()). The result is multiplied by -1 if entry.getValue().equals("desc").

Lab4

Simpledb/materialize/SortPlan.java	Modify runs iteration to > 1.
Simpledb/opt/TablePlanner.java	Create a makeMergeJoin method that calls MergeJoinPlan and passes to it the current plan, myplan, and the respective field names. Call makeMergeJoin in makeJoinPlan if p is null when makeIndexJoin is assigned to Plan p.
Simpledb/materialize/MergeJoinPlan.java	Modify List<String> type in MergeJoinPlan to List<AbstractMap.SimpleEntry<String,String>>.

Lab5 onwards

Simpledb/parse/Parser.java	Create method groupList() to parse group by preds and add it to groups list. Modify parser regarding 'select' to determine if keyword is aggregate or just normal identifiers or '*' for select all. Add groups and aggregates to QueryData constructor create method newAggregationFn() to parse aggregate and return the specific aggregation function with the field involved. Support for distinct by checking in normal selection and in aggregation fn
Simpledb/parse/Lexer.java	Create matchAggregate() to return true if the current word is an aggregate. Create eatAggregate() Initialise aggregates list with initAggregates() using 'avg', 'sum', 'min', 'max' and 'count'
Simpledb/parse/QueryData.java	Initialise aggregates, aggregatesFields and groups. Create methods to return their values.
Simpledb/opt/HeuristicQueryPlanner.java	Add GroupByPlan in between SortPlan and ProjectPlan. GroupByPlan activates only when aggregates or groups is not empty.

Simpledb/plan/ProjectPlan.java	Modify constructor initialisation to use schema.addAll if select ‘*’ is applied.
Simpledb/materialize/AvgFn.java Simpledb/materialize/SumFn.java Simpledb/materialize/MinFn.java	Create this three classes which implements AggregationFn and their individual logic
Simpledb/materialize/AggregationFn.java	Add a new method field() to obtain the field which the aggregate is working on.
Simpledb/materialize/HashJoinPlan.java	Create this plan to support hash join. partition size is determined by tx.availableBuffers()-1. blocksAccessed() is 3*(p1.blocksAccessed() + p2*blocksAccessed()) open() returns HashJoinScan which requires the two scans, two joinfields, partitionSize and current schema’s fields.
Simpledb/materialize/HashJoinScan.java	First, create a hashMap and limit its range to the partitionSize to simulate real scenario. For each LHS record, copy this record and add it to the hash table based on the hash value of the joinfield. Next, for each RHS record, get the partition number by hashing the joinfield. If it matches any partition, return true for this RHS record with each and every LHS record in that partition. Modify all the getXX(String fldname) implementation to obtain the LHS values by reading from the current partition.
Simpledb/opt/TablePlanner.java	add in support for hash join