CS 564 Assignment 6
Minirel Query and Update Operators Discussion

Overview

3. Buffer Manager (Done)	**4. HeapFile Manager**
~~5. Front-End and Database Utilities~~ (Skipped)	**6. Query and Update Operators (We're here!)**

Stage 5

How Minirel works?
+ dbcreate
+ minirel
+ dbdestroy

Stage 5

Minirel: runs a loop
+ show prompt
+ get a user command
+ call parse() to parse it into an internal format
+ call interp() to understand what the query wants to do, then call the appropriate backend procedure
+ show results

Stage 5

Implement the front-end and database utilities of Minirel, including catalog relations

Front-End Command Syntax

- **<DDL_STATEMENT> ::= <CREATE relation>**
  | <DESTROY>
  | <LOAD>
  | <PRINT>
  | <HELP>
  | <QUIT>

Stage 5

Implement the front-end and database utilities of Minirel, including catalog relations

Includes

- parser: parse user commands and SQL
- dbcreate: creates a unix directory to hold the database; creates database catalogs
- dbdestroy: destroys the database
- minirel: creates a buffermanager, opens the relation and attribute catalogs and then calls parse()

FAQ

Minirel Commands

- DDL (Data Definition Language): perform various utility operations
    - create table, destroy table, load table, print table, help, quit
- DML (Data Manipulation Language): answer queries
    - query or update

- query: select ... from ... where ... (can do selection or join)
- update: delete from table where ..., insert into table ...

Stage 5

Implement the front-end and database utilities of Minirel, including catalog relations

Two tables that form the catalog:
- relation catalog (RelCatalog): one tuple for each relation (including itself)
- attribute catalog (AttrCatalog): one tuple for each attribute of every relation

Both are created by the dbcreate utility and together they contain the schema of the database.

Stage 5

Stage 5:
dbcreate, dbdestroy, backend procedures to support Data Definition Language

Stage 6:
backend procedures to support Data Manipulation commands
- selection, insertion, deletion, NO JOIN

Stage 6

Stage6 TODOs:

- select.C
    - QU_Select(...), ScanSelect(...)

- insert.C
    - QU_Insert(...)

- delete.C
    - QU_Delete(...)

QU_Select

```
QU_Select(const string & result,
          const int projCnt,
          const attrInfo projNames[],
          const attrInfo *attr,
          const Operator op,
          const char *attrValue)
```

QU_Select

+ Make sure to give ScanSelect the proper input

+ To go from attrInfo to attrDesc, need to consult the catalog (attrCat and relCat, global variables)
    + go through the projection list and look up each in the attr catalog to get an AttrDesc structure (for offset, length, etc)

ScanSelect

```
ScanSelect(const string & result, ==> table to store output
           const int projCnt,
           const AttrDesc projNames[],
           const AttrDesc *attrDesc, ==> attr for selection
           const Operator op,
           const char *filter, ==> *attrValue
           const int reclen)
```

ScanSelect

+ have a temporary record for output table
+ open "result" as an InsertFileScan object
+ open current table (to be scanned) as a HeapFileScan object
+ check if an unconditional scan is required
+ check attrType: INTEGER, FLOAT, STRING
+ scan the current table
+ if find a record, then copy stuff over to the temporary record (memcpy)
+ insert into the output table

QU_Delete

```
QU_Delete(const string & relation,
          const string & attrName,
          const Operator op,
          const Datatype type,
          const char *attrValue)
```

QU_Delete

+ when an input argument is NULL
    + if attrName is NULL, set startScan's offset and length to 0, type to string, filter to NULL.

+ scan depends on attr type: INTEGER/FLOAT/STRING

QU_Insert

```
QU_Insert(const string & relation,
          const int attrCnt,
          const attrInfo attrList[])
```

QU_Insert

+ Insert a tuple with the given attribute values (in attrList) in relation. The value of the attribute is supplied in the attrValue member of the attrInfo structure.

+ Make sure that attrCnt corresponds the relation attribute count
    + Otherwise, return OK

+ Be careful with the order of attributes

+ Use InsertFileScan::insertRecord()

+ Due May 1, 11:59pm

+ Check out Stage 5 before you start

Questions