The Front End (FE) Layer: System Report

Rut Diane Cuebas (2019-28748)
Jungwoo Yang (2019-23417)

1. Environment
   Ubuntu and Windows WSL

2. Compile Method
   - make
   - have to set directory accordingly

3. File Structure
   fe/fe.c – contains implementations for utility functions (i.e. CreateTable, BuildIndex) and query functions (i.e. select, join, insert).

   h/fe.h – contains prototypes for functions defined in fe.c and defines the datatype structures ATTR_DESCR, REL_ATTR, and ATTR_VAL, that define attributes.

   fe/feinternal.c – contains implementations for catalog search functions, attribute search functions, and other helper functions for the fe layer.

   fe/feinternal.h – contains prototypes for functions defined in hfinternal.c

   h/catalog.h – defines the structures for a relcat relation, and an attract relation

4. Data Structures

   relcat

   A catalog (table) that is used to keep track of all of the relations in the database. The structure of each tuple is defined below.

```c
/* structures of relcat relation */

typedef struct _relation_desc {
    char relname[MAXNAME];  /* relation name            */
    int  relwid;        /* tuple width (in bytes)      */
    int  attrcnt;       /* number of attributes        */
    int  indexcnt;      /* number of indexed attributes    */
    char primattr[MAXNAME]; /* name of primary attribute       */
} RELDESCTYPE;
```

   attrcat

   A catalog that contains all of the attributes of all of the relations in the database. The structure of each tuple is defined below.

```c
/* structure of attrcat relation */
```

```
2.
3.  typedef struct _attribute_desc {
4.      char relname[MAXNAME];  /* relation name            */
5.      char attrname[MAXNAME]; /* attribute name           */
6.      int  offset;        /* attribute offset in tuple        */
7.      int  attrlen;       /* attribute length         */
8.      int  attrtype;      /* attribute type           */
9.      bool_t  indexed;        /* if the field is indexed or not       */
10.     int  attrno;        /* attr number (used in index name) */
11. } ATTRDESCTYPE;
```

5. Implementations

feinternal.c/findRelcat

- Finds **relcat** using a file scan.
- Sets **relrecidPtr** to the record id of the **relcat** table's first record.

feinternal.c/findAttrcat

- Finds **attrcat** using a file scan.
- Sets **attrrecidPtr** to the record id of the **attrcat** table's first record.

feinternal.c/getAttrcat

- Returns the **attrcat** of the specified relation name.
- Uses a file scan to find **attrcat** and loops through all records, copying them to the location of **attrStrArrPtr**.

feinternal.c/findAttrVal

- Iterates through the values of an attribute to find any attribute that matches the given **attrName** and returns its value**.**

feinternal.c/findAttrName

- Iterates through the attributes of the relation to find the matching attribute name.

feinternal.c/openHFAM

- Iterates through the attributes of the relation, if the attribute is indexed it adds it to the AM layer index table.

feinternal.c/closeHFAM

- Iterates through the attributes of the relation, if the attribute is indexed, it removes it from the AM layer index table.

feinternal.c/openHFAMScan

- Opens a HF file scan
- Opens an AM index scan for only indexed attributes

feinternal.c/closeHFAMScan

- Closes an HF file scan and AM index scan

feinternal.c/UpdateAttrVal

- Updates the **ATTR_VAL** attributes of **attrVal**

feinternal.c/updateAttrValJoin

- Updates the **ATTR_VAL** attributes in the case of a join