

Minirel Part-2

Utilities and Queries

...And then there was one.

EECS 484
Winter 09

Design the whole System!

- Create tables
- Insert records into tables
- Query the tables

This time, we give you the Buffer Manager!

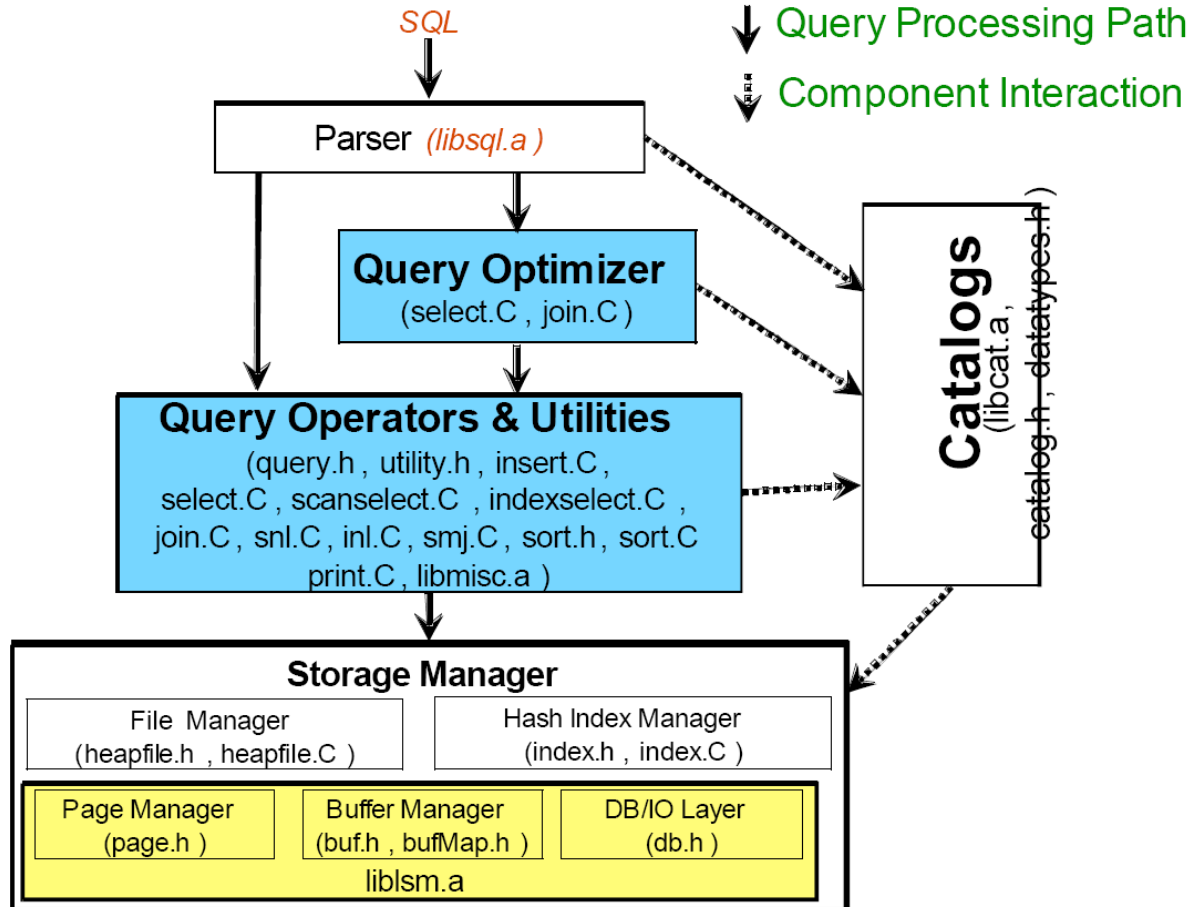
... sort of

SELECT sid, course FROM Students



<u>sid</u>	<u>course</u>
111	DBMS
112	OS

Minirel Architecture



You have...

- Sort
- Hash Indexing
- Buffer Manager
- Relation and Attribute Catalog definitions

Catalogs

- Meta data
- i.e. info regarding attributes and relations
- How would they be stored??

- Relations!

Classes for Catalogs

- **RelDesc: attributes, indexes on a relation**

```
typedef struct {
    char relName[MAXNAME]; // relation name
    int attrCnt;           // number of attributes
    int indexCnt;         // number of indexed attrs
} RelDesc;

class RelCatalog : public HeapFileScan {
public:

    // get relation descriptor for a relation
    const Status getInfo(const string & rName, RelDesc& record);

    // add information to catalog
    const Status addInfo(RelDesc & record);
...

```

Classes for Catalogs

- AttrDesc: Attribute Description

```
typedef struct {
    char relName[MAXNAME]; // relation name
    char attrName[MAXNAME]; // attribute name
    int attrOffset; // attribute offset
    int attrType; // attribute type
    int attrLen; // attribute length
    int indexed; // TRUE if indexed
} AttrDesc;

class AttrCatalog : public HeapFileScan {
public:
    // get attribute catalog tuple
    const Status getInfo(const string & rName, const string & attrName, AttrDesc
        &record);

    // add information to catalog
    const Status addInfo(AttrDesc & record);

    // get all attributes of a relation
    const Status getRelInfo(const string & rName, int &attrCnt, AttrDesc *&attrs);

```

...

'Utilities'

- CREATE provided
- Your job: INSERT
- Some issues to consider
 - The signature is provided
 - Reject if all attributes not specified
 - Attributes may be in any order
 - Update index


```
class HeapFile {
protected:

    File*      file;
    HeaderPage* headerPage;
    int headerPageNo;

public:
    // initialize
    HeapFile(const string & name, Status& returnStatus);

    // insert record into file
    const Status insertRecord(const Record & rec, RID& outRid);
```

Select Queries

- The “WHERE relation.attr > 2” kind
- 2 access methods
 - File Scan
 - Index Scan
- “Optimization”
 - Hash Indexing used
 - So index scan, when index and equality
- Issues
 - No WHERE clause, then attribute list is NULL
 - Result stored in *result* relation

Select Queries

```
static Status Select(const string & result,  
                    // name of the output relation  
                    const int projCnt,  
                    // number of attributes in the projection  
                    const attrInfo projNames[],  
                    // the list of projection attributes  
                    const attrInfo *attr,  
                    // attribute used in the selection predicate  
                    const Operator op,  
                    // predicate operation  
                    const void *attrValue);  
                    // literal value in the predicate
```

Join Queries

- The “WHERE rel1.attr1 = rel2.attr2” kind
- 3 join algorithms
 - Index on either join attribute: Index Nested Loop
 - Not equi-join: Simple Nested Loop
 - Equi-join, neither indexed: Sort Merge

Sort Merge

- $k = 80\%$ of available buffers
- $n =$ number of tuples that can fit on the k buffers
- Use Heapfilescan, storing n tuples at a time
- Sort tuples, store in temporary files [runs]
- Merge all temporary files, upto k Heapfilescans
- Issues
 - Duplicates [back and forth]
 - Result stored in *result*

Bootstrapping Catalogs

- *attrcat* and *relcat* are relations
- You must add data for themselves into the tables
- 2 records into *relcat*
- A record for each attribute of the relations into *attrcat* [9 = 3 + 6]

Catalogs

Explanatory example: Attribute_Cat

attrName	relName	type	position
attrname	Attribute_Cat	string	1
relname	Attribute_Cat	string	2
type	Attribute_Cat	string	3
position	Attribute_Cat	integer	4
sid	Students	string	1
name	Students	string	2
login	Students	string	3
age	Students	integer	4
gpa	Students	real	5
fid	Faculty	string	1
fname	Faculty	string	2
sal	Faculty	real	3

Checklist

1. **insert.cpp**: Inserting tuples
2. **query.h**: Selects and the joins
3. **select.cpp**: Call the appropriate select algorithm.
4. **scansselect.cpp**: Select using a file scan
5. **indexselect.cpp**: Index selection
6. **join.cpp**: Pick the appropriate join algorithm
7. **snl.cpp**: Simple nested loop algorithm.
8. **inl.cpp**: Index nested loops algorithm
9. **smj.cpp**: sort merge join
10. **dbcreate.cpp**: Bootstrapping the catalogs

Use sort.h,
sort.cpp



Good Luck!