ECS 165B: Database System Implementation
Lecture 17

UC Davis
May 5, 2010
Class Agenda

• Last time:
  – Introduction to XML

• Today:
  – Overview of DavisDB Part 3: System Manager
  – Mid-course evaluation

• Reading:
  – none
Announcements

DavisDB Part 3 due Tuesday 5/11 @11:59pm

Necessary files already in your repositories
DavisDB, Part 3: System Manager
Your Mission, Should You Choose to Accept It

• We provide a command-line shell interface (SystemParser)
  – create / drop / open / close database
  – create / drop / bulk-load / print tables
  – create / drop indices
  – list relations in database, information about their attributes

• You implement the backing functionality (SystemManager)
  – the operations above (using RecordManager and IndexManager)
  – plus, system catalogs
DavisDB System Catalogs

- Two system catalogs – relations and attributes – maintain metadata about tables, their attributes, and indices
- Catalogs are tables too; create using RecordManager
- Exact contents of these tables is up to you! E.g.,

<table>
<thead>
<tr>
<th>relations</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>relName</td>
<td>relName Name of the relation</td>
</tr>
<tr>
<td>recordSize</td>
<td>attrName Name of the attribute</td>
</tr>
<tr>
<td>nAttributes</td>
<td>attrNo Number of the attribute</td>
</tr>
<tr>
<td></td>
<td>attrType Type of the attribute</td>
</tr>
<tr>
<td></td>
<td>attrLength Length of the attribute in bytes</td>
</tr>
<tr>
<td></td>
<td>offset Offset from beginning of record</td>
</tr>
<tr>
<td></td>
<td>hasIndex Has a B+ tree index?</td>
</tr>
</tbody>
</table>
DavisDB System Catalogs (2)

• Catalog tables should include information about the catalog tables themselves!

• So, even if database has no other tables:
  
  – relations table should have two tuples (one for relations, one for attributes)
  
  – attributes table should have rows for the attributes in catalog tables
The Command-Line Shell

quatchi:src green$ ./DavisDB
Welcome to the DavisDB command-line shell.
Type "help;" for help.
The Command-Line Shell: Commands

create database <dbName> ;
drop database <dbName> ;
open database <dbName> ;
close database ;
create table <tableName> (<attrName> <attrType>, ..., <attrName> <attrType>) ;
   where <attrType> is either float, int, or char(<length>)
drop table <tableName> ;
create index <tableName> (<attrName>) ;
drop index <tableName (<attrName>) ;
load <tableName> <fileName> ;
info ;
info <tableName> ;
print <tableName> ;

(io print ;
io reset ;
help ;
quit ;  (implemented by corresponding SystemManager methods)

(print or reset page I/O statistics) (built-in shell commands)
create database <dbName> ↔ SystemManager::createDb

- Database should be created in a new subdirectory dbName

  1. mkdir(dbName)
  2. chdir(dbName)
  3. create system catalogs
  4. chdir("..")

- Useful Linux library commands: mkdir, chdir
  - #include <sys/stat.h>
open database <dbName> ⇔ SystemManager::openDb

1. chdir(dbName)

2. Open system catalogs and keep them open (they will be used heavily)
close database ⇔ SystemManager::closeDb

1. Close system catalog tables

2. chdir("..")
drop database <dbName> ⇔
SystemManager::dropDb

1. system("rm -rf <dbName>")
   – Need to #include <stdlib.h>
create table <relName>(...) ⇐
SystemManager::createTable

1. Create the specified table (RecordManager)

2. Update system catalogs to record information about table and its attribute
create index <relName> (<attrName>) ↔ SystemManager::createIndex

1. Open table (RecordManager)

2. Create a new B+ tree index (IndexManager)

3. For each record in table (RecordFileScan)
   1. Insert key, recordID into index

4. Update system catalogs to reflect existence of index
load <relName> <fileName> ⇔ SystemManager::load

1. Bulk-load the contents of comma-separated file into table
2. For each line in file:
   1. Parse the line, create a record
   2. Insert record into table
   3. Insert keys for record into any indices

• How to read the file:
  (C) #include <stdio.h>; use fopen, getc*, fclose
  (C++) #include <fstream>; use ifstream::open, ifstream::getline*, ifstream::close

• How to parse a line:
  – lines look like "22,Dustin,7,45.0" – no commas inside strings
  – use strtok (<string.h>), strtol and strtol (<stdlib.h>)
print <relName> ⇔ SystemManager::print

1. Open the table (RecordManager)

2. Open a scan on the table (RecordFileScan)

3. Call SystemPrinter::printHeader to print table header

4. For each record in table
   1. Call SystemPrinter::printRecord to print record

5. Call SystemPrinter::printFooter to print table footer

6. Close scan and table

• SystemPrinter is provided for you, don't change
print: Sample Output

quatchi:src green$ ./DavisDB
Welcome to the DavisDB command-line shell.
Type "help;" for help.
open database Boathouse;
  => RC_OK
print Boats;
Boats.bid int, Boats.bname char(32), Boats.color char(16))
-------------------------------------------------------------
101,Interlake,blue
102,Interlake,red
103,Clipper,green
104,Marine,red
4 records total.
  => RC_OK
Testing: Boathouse.ddb and Sailors.csv, Boats.csv, Reserves.csv

• Boathouse.ddb: a sample DavisDB session exercising most of the SystemManager functionality

  quatchi:src green$ ./DavisDB < Boathouse.ddb

• Uses load to populate tables with data from *.csv

• Make your own tests too!
Debugging Strategies

• Many debuggers (e.g., Eclipse) don't support debugging console applications very well
  – UI debugging in general can be problematic!

• Script files like Boathouse.ddb can be extremely useful
  – e.g., in gdb, use "run params < Boathouse.ddb" to pipe input from the file
Guidelines and Tips

• Make plenty of helper functions to access system catalogs for specific relations and/or attributes!
  – These will be needed now (for code cleanliness) and later (in Part 4)

• Don't forget to insert descriptions for attributes and relations into attributes and relations

• Use RecordFileHandle::forceAllPages when you modify the catalog tables
  – To avoid seeing stale data when you do print attributes or print relations

• Don't forget to update indices when you update tables

• Should allow creating indices on catalog tables; should disallow drop or load of catalog tables
Guidelines and Tips (2)

• This part of the project is definitely easier than Parts 1 and 2, but still takes a while

• Don't put it off to the last minute!