Programming in Perl

Week Thirteen

Programming Databases

Flat File Databases

- Flat file database are one or more files that we directly access

```
open FILE, ">foo.dat" or die "Opps! Can't open!";
while (<FILE>) {
    ...
}
```

- Easy to understand
  - We define the format
  - We control when the file is accessed
- Flat file DB's are usually application specific
- It's possible to do very complex databases using multiple files and the low level file I/O calls

Low Level File I/O

- `read FILEHANDLE, $buffer, $length, $offset`
  - Read a specified number of bytes from the file handle and put them into a scalar variable
  - The number of bytes actually read is returned
  - If an offset is specified, the bytes are placed at that offset in the scalar variable
- `seek FILEHANDLE, $position, $whence`
  - Move the current position in a file
  - If `$whence` is:
    - `0`, move from the start of the file
    - `1`, move from the current position
    - `2`, move from the end of the file

Low Level File I/O

- Two ways to use low level I/O for DB's
  - Fixed size records
    - Easy to locate, read, and write a record

```
seek FILE, $record_number * $record_size, 0;
read FILE, $buffer, $record_size;
```

    - Fixed size records are wasteful of space
  - Variable sized records
    - One implementation is to separate each record with an "End-Of-Record" marker, like `\n`
    - Another is to write the length of the record as the first data of the record
    - Variable sized records use space effectively in a file
    - Variable sized records make it difficult to locate a specific record in a file

Simple Database Interface

- The problem with flat file databases is that you must create your own access routine, and worry about the physical format of the data
- The Berkeley Database is a simple DB that has a interface library that handles most of the work for you
- There is a Perl interface called `DB_FILE.pm`
- The Berkeley DB has three database formats:
  - `DB_HASH`- A hash like DB
  - `DB_BTREE`- An ordered hash like DB
  - `DB_RECNO`- A record based DB

Using DB_File.pm

```
use DB_File;

my %hash;
tie %hash, 'DB_File', 'database' or die "Can't open database: $!\n";
%hash{'apple'} = 'red';
%hash{'orange'} = 'orange';
%hash{'banana'} = 'yellow';

print "Banana's are $hash{'banana'}\n";

foreach my $key (keys %hash) {
    print "$key -> $hash{$key}\n";
}
untie %hash;
```

Tied Variables

- The `DB_File` module uses the general `tie()` interface
- `tie()` binds a variable to a class that provides access methods for the variable
- The general interface is:

```
tie VARIABLE, CLASSNAME, LIST;
```

- You can tie may types of variables: scalar, array, hash, file handle

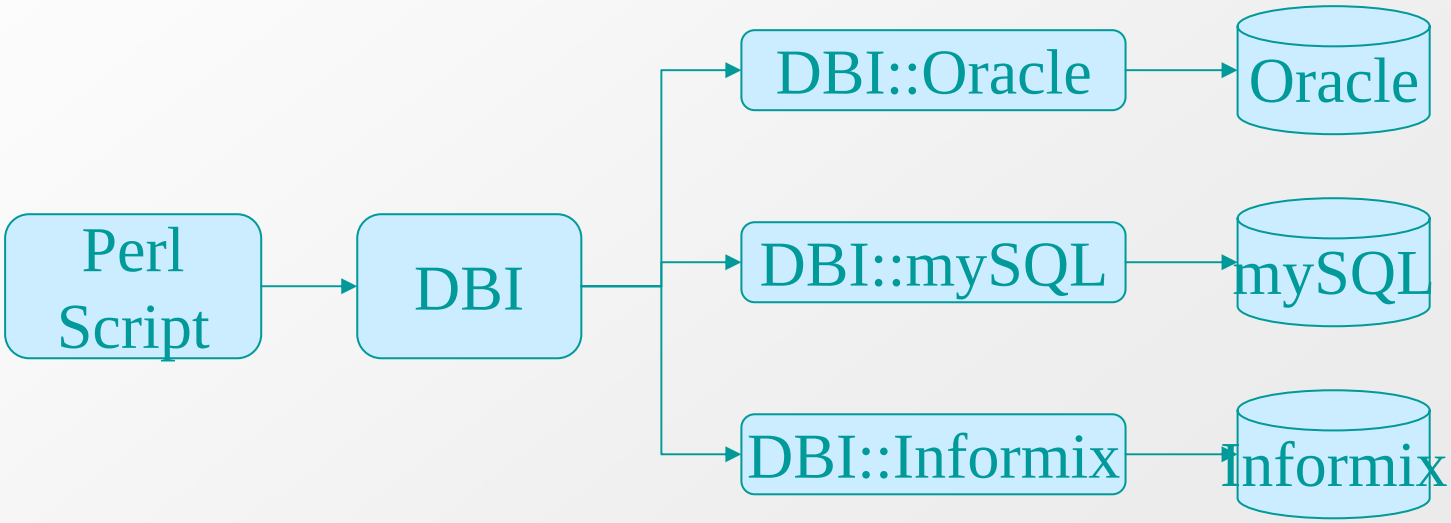
Tying a hash

- For a hash, you would provide several methods:

```
TIEHASH classname, LIST
FETCH this, key
STORE this, key
DELETE this, key
CLEAR this
EXISTS this, key
FIRSTKEY this
NEXTKEY this
DESTROY this
UNTIE this
```

The Perl DBI

- You can also work with Relational Databases like Oracle, MySQL, SQLServer, etc.
- `DBI.pm` is a set of generalized interfaces to Relational Databases



DBI Example

```
use DBI::Oracle;

my $dbh = DBI->connect( "dbi:Oracle:archaeo", "username",
   "password" );

my $sth = $dbh->prepare( "SELECT id, name FROM megaliths" );
$sth->execute();
while ( ($name) = $sth->fetchrow_array) {
    print "$name\n";
}
```