

Exam 3 Answers:

1. Relational databases store data in tables and provide a method for relating columns in one table to columns in another to create larger 'virtual' tables on which queries can be performed. In this way the actual information stored in the database need not be replicated in multiple rows of a table that is actually stored in the system.

2. The Regular Expression:

`^d\D+(\d+)\s+\d+\s+\d+\s+\d+\s+(\D+)/`

will match:

- "1 T, 2000 0 0 4 R"
- "2 A, 200 0 0 3 R"
- "3 G, 1000 100 12 1 Tele"

placing the first number past the comma in \$1 and the last word in \$2.

3. Creating the database could be performed manually, or using a slightly modified version of the compound.sql file. This can be loaded into the query window of SQLyog, edited and then executed. The perl program db2sql.pl must also be modified with your username and password.

4.a

```
select count(mol_id) from molecule;
```

4.b.

```
select count(atom_type)/count(distinct mol_id) from atom where atom_type not like '%H%';
```

4.c. Without the ability to nest select statements, or enforced referential integrity, this cannot be done in one SQL statement. If referential integrity were enforced, and if name were a child of molecule, then deleting the record from molecule where a join between molecule and name were performed and where name like 'BENZ%' were performed, then the 'ON DELETE CASCADE' would delete the records in all child tables.

For the exam, you can get full credit for any statement which deletes records in either name or molecule based on the LIKE 'BENZ%' statement. (Next time we will use MySQL 4 and foreign keys...).

4.d. This amounts to inserting a new record into name where the mol_id is the same as the acetone mol_id, unfortunately, you cannot get this in one step either. Any statement which inserts something into name properly is given full points.

5. Several people submitted excellent versions of this program the following is from Brian Laughlin:

```
#!/usr/local/bin/perl

use DBI;

my $dsn = "DBI:mysql:laughlin:miner.chem.purdue.edu"; # data source
my $user_name = "blaughlin"; # user name
my $password = "laughlib"; # password

# connect to database
my $dbh = DBI->connect( $dsn, $user_name, $password,
    { RaiseError => 1, PrintError => 0 } );

# begin loop to fill database - get no. of rows

$sth = $dbh->prepare(qq{select count(mol_id) from molecule});
$sth->execute();

my @mol_rows = $sth->fetchrow_array();
my $mol_rows = $mol_rows[0];

for( $i = 1; $i<= $mol_rows; $i++ )
{
# get no. of bonds

    $sth = $dbh->prepare(qq{
        select sum(bond_type)
        from bond
        where mol_id = $i});
    $sth->execute();

    my @ary = $sth->fetchrow_array();
    my $bonds = $ary[0];

# get atom types

    $sth = $dbh->prepare(qq{
        select atom_type
        from atom
        where mol_id = $i});
    $sth->execute();

    my $valence = 0;
    my $ave_mw = 0;
    my $exact_mw = 0;

    my @atom;

    while(@atom = $sth->fetchrow_array)
    {
        $atoms = "$atoms $atom[0]";

        $sth_a = $dbh->prepare(qq{
            select valence, ave_mass, exact_mass
```

```

        from element
        where atom_type = "$atom[0]");
        $sth_a->execute();

        my @element;
        while(@element = $sth_a->fetchrow_array)
        {
            $valence = $valence + $element[0];
            $save_mw = $save_mw + $element[1];
            $exact_mw = $exact_mw + $element[2];
        }

        $sth_a->finish ();
    }

# calc no. of hydrogens

    my $hyd = $valence - ( $bonds*2);

# get H masses

    $sth = $dbh->prepare(qq{
        select ave_mass, exact_mass
        from element
        where atom_type = 'H'});
    $sth->execute();

    my @hydrogen = $sth->fetchrow_array;
    $save_hyd = $hydrogen[0];
    $exact_hyd = $hydrogen[1];

#calc final MW adding hydrogen masses

    $save_mw = $save_mw + ($save_hyd * $hyd);
    $exact_mw = $exact_mw + ($exact_hyd * $hyd);

#update database

    $sth = $dbh->prepare(qq{
        UPDATE molecule SET ave_mw = $save_mw, exact_mw = $exact_mw
        WHERE mol_id = $i});
    $sth->execute();

    print ".";
}

# finish program

$sth->finish();
$dbh->disconnect();
exit(0)

```

6. Again several people submitted excellent versions of this program, here is Hao Xu's program:

```
#!/usr/local/bin/perl

use DBI;

$dsn = "DBI:mysql:xu:miner.chem.purdue.edu";    # data source
$user_name = "hxu";                            # user name
$password = "xu23";                            # password

$file='aveMW.txt';
$data="Average_MW";

# connect to database
$dbh = DBI->connect( $dsn, $user_name, $password,
                    { RaiseError => 1, PrintError => 0 } );
$sth = $dbh->prepare(qq{
    SELECT $data
    FROM molecule
    WHERE Average_MW > 0
    ORDER BY $data});
$sth->execute();

for( $i = 0; $i<= 732; $i++ )
{
    @ary = $sth->fetchrow_array();
    @aveMW[$i]=$ary[0];
}

$sth->finish();
$dbh->disconnect();
open OUTPUT_FILE, ">$file";
print OUTPUT_FILE "@aveMW\n";
close(OUTPUT_FILE);

system ("C:\\Progra-1\\R\\bin\\Rterm.exe --no-save <plot.r");

exit(0);
```

Where the R program looked like this:

```
setwd("d:\\hao\\study\\chm696d\\exam3")
averageMW<-read.table("aveMW.txt")
averageMW<-t(averageMW)
pdf(file="Histogram.pdf")
hist(averageMW,br=50)
dev.off()
```

And the output (from the pdf file):

Histogram of averageMW

